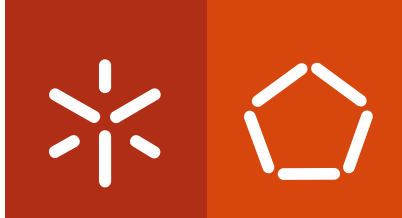


Universidade do Minho  
Escola de Engenharia

Fábio Raul da Costa Gonçalves

Intelligent Intrusion Detection  
System for Vehicular Ad hoc  
Networks





**Universidade do Minho**

Escola de Engenharia

Fábio Raul da Costa Gonçalves

Intelligent Intrusion Detection System for  
Vehicular Ad hoc Networks

**Programa de Doutoramento em Informática  
das Universidades do Minho, de Aveiro e do Porto**



Universidade do Minho

Work Supervised by

Professor Joaquim Melo Henriques Macedo

Professor Alexandre Júlio Teixeira Santos

February 2022

# Copyright

This is an academic work that can be used by third parties as long as the internationally accepted rules and good practices are respected with regard to copyright and related rights. Thus, the present work may be used under the terms set out in the license below.



CC BY

<https://creativecommons.org/licenses/by/4.0/>

# **STATEMENT OF INTEGRITY**

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

# Sistema de Detecção de Intrusões Inteligente para Redes Veiculares Ad hoc

Intelligent Transportation Systems (ITS) é um conjunto de aplicações e serviços que têm como objetivo tornar a condução mais fácil e segura sem menosprezar a segurança de dados ou a privacidade. Estas utilizam comunicações para, de forma cooperativa, melhorar ou introduzir novas funcionalidades. As Vehicular Ad hoc Networks (VANETs) oferecem um meio para comunicação entre os diversos nós. No entanto, estas têm características desafiadoras, principalmente no que toca a segurança.

Assim, a segurança em VANETs é tema de diversos trabalhos de investigação, a maioria baseados em criptografia e que têm como objetivo prevenir e/ou mitigar ataques. No entanto, alguns dos ataques, como os realizados por entidades fidedignas ou que têm como alvo as vulnerabilidades das ferramentas criptográficas, não são facilmente preveníveis.

Os Intrusion Detection Systems (IDSs) utilizam uma estratégia diferente. O seu objetivo não é prevenir ataques mas detetá-los de forma a despoletar uma resposta de forma minimizar os efeitos causados no sistema alvo. Este trabalho de Doutoramento tem como objetivo o desenho e a validação de um IDS capaz de detetar ataques numa VANET. O IDS está estruturado numa hierarquia com diversos clusters em cada nível, divididos de acordo com as necessidades e características de cada nó. Esta divisão habilita o uso de ferramentas de Machine Learning (ML) e a sua adaptação às características e capacidades dos nós de cada cluster. A comunicação entre nós deverá utilizar mecanismos de segurança fortes com o âmbito de proteger os dados trocados.

Os resultados obtidos indicam que a arquitetura proposta é capaz de detetar múltiplos ataques com sucesso, e permite aos diversos clusters utilizar o algoritmo de ML que melhor responda às suas necessidades. Os nós de níveis mais altos do IDS utilizam algoritmos mais complexos e que permitem melhores deteções à custa de necessidade de mais Central Processing Unit (CPU) e tempos de decisão mais longos. Por outro lado, os nós em níveis mais baixo da hierarquia utilizam algoritmos mais leves, permitindo decisões rápidas com menor uso de CPU, mas com taxas de deteção mais baixas.

**Palavras Chave:** VANETs, Segurança, Sistemas de Detecção de Intrusão, Machine Learning

# Intelligent Intrusion Detection System for Vehicular Ad hoc Networks

Intelligent Transportation Systems (ITS) are a set of applications and services that aims to make driving easier and safer, without neglecting security or user privacy. These applications take advantage of communications to, cooperatively, improve their capabilities or provide new functionalities. Vehicular Ad hoc Networks (VANETs) provide a communication medium to allow the multiple nodes to communicate. Nevertheless, these possess specific characteristics that make them quite challenging, particularly regarding security.

Thus, VANET security is a subject of research, resulting in multiple published research works. Most of them use the cryptographic approach that aims to prevent and/or mitigate attacks. However, some attacks, such as those performed by authentic entities or targeting the vulnerabilities of the cryptographic tools, cannot be easily prevented.

Intrusion Detection Systems (IDSs) use a different strategy. Their goal is not to prevent attacks but detect them and trigger a response to minimize the effects on the targeted system. This Ph.D. work aims to design and validate an Intelligent IDS to detect attacks in VANETs. The IDS is structured into a hierarchy with multiple clusters by level, according to the characteristics and needs of each node. Thus, enabling the usage of Machine Learning (ML) algorithms and adapting their characteristics and capabilities to the cluster's nodes. The communication between the nodes should use strong security mechanisms to protect the data exchanged.

The results indicate that the proposed architecture is able to detect multiple attacks accurately, and enables each cluster to use the ML algorithms that better respond to its needs. The higher levels of the IDS use more complex algorithms that allow better detections at the cost of more Central Processing Unit (CPU) and longer detection times. On the other hand, the lower levels use light algorithms that allow quick and CPU light operations but have lower detection rates.

**Keywords:** VANETs, Security, Intrusion Detection Systems, Machine Learning

# Contents

<b>Resumo</b>	<b>iii</b>
<b>Abstract</b>	<b>iv</b>
<b>Contents</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>List of Figures</b>	<b>xiv</b>
<b>Acronyms</b>	<b>xviii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Objectives . . . . .	2
1.2 Methodology of the Research . . . . .	4
1.3 Scientific Contributions . . . . .	6
1.4 Organization of the Document . . . . .	7
<b>2 Background</b>	<b>11</b>
2.1 Security . . . . .	11
2.1.1 Cryptography . . . . .	12
2.1.2 Public Key Infrastructure . . . . .	15
2.1.3 Certificate Revocation Lists . . . . .	15
2.1.4 Attribute Based Encryption . . . . .	16



2.1.5	Identity Manager . . . . .	16
2.2	Vehicular Ad hoc Networks . . . . .	17
2.2.1	Characteristics . . . . .	18
2.2.2	VANET Architectures . . . . .	19
2.2.3	Medium Access Standards . . . . .	22
2.3	VANET Security Requirements . . . . .	24
2.4	VANET Attacks . . . . .	25
2.4.1	Types of Attacks . . . . .	25
2.4.2	Cryptographic Solutions . . . . .	27
2.5	Security Models for VANETS . . . . .	29
2.5.1	Based on Public Key Infrastructure . . . . .	29
2.5.2	Identity-Based Encryption . . . . .	30
2.5.3	Based on Situation-modeling . . . . .	31
2.6	Intrusion Detection Systems . . . . .	31
2.7	Machine Learning . . . . .	33
2.7.1	Supervised Learning . . . . .	33
2.7.2	Unsupervised Learning . . . . .	34
2.8	Platooning . . . . .	35
2.8.1	Platooning Maneuvers . . . . .	36
2.8.2	Platooning Security Requirements . . . . .	37
<b>3</b>	<b>Related Work</b>	<b>40</b>
3.1	Systematic Literature Review Methodology . . . . .	40
3.1.1	Research Questions . . . . .	41
3.1.2	Literature Sources and Search String . . . . .	42
3.1.3	Studies selection . . . . .	43
3.1.4	Inclusion/Exclusion Criteria . . . . .	44
3.1.5	Data Extraction . . . . .	45

3.2	Synthesis of Collected Data in the SLR . . . . .	47
3.3	Evolution of the ML-Based Research Work . . . . .	56
3.4	Analysis of Experimental Results . . . . .	58
<b>4</b>	<b>Securing Communications</b>	<b>60</b>
4.1	VPKIbrID Message Format . . . . .	61
4.2	VPKIbrID Cipher Modes . . . . .	62
4.2.1	VPKIbrID Public Key Infrastructure . . . . .	63
4.2.2	VPKIbrID Attribute-Based Encryption . . . . .	64
4.3	VPKIbrID Interactions . . . . .	65
4.3.1	Obtaining Identity Manager Token . . . . .	66
4.3.2	Obtaining Pseudonym Certificates . . . . .	67
4.3.3	Obtaining Attribute Base Encryption Keys . . . . .	68
4.3.4	User Authentication . . . . .	69
4.4	Secure Platooning . . . . .	71
4.4.1	Platoon Creation . . . . .	72
4.4.2	Initialization . . . . .	73
4.4.3	Disseminate Platooning Advertisements . . . . .	73
4.4.4	Joining Platoon . . . . .	74
4.4.5	Secure Message Exchange . . . . .	74
<b>5</b>	<b>Datasets Synthesis with Security Threats</b>	<b>76</b>
5.1	Scenarios for Message Generation . . . . .	77
5.2	Geographical Maps for Message Collection . . . . .	78
5.3	Simulation Setup . . . . .	80
5.4	Data Collection . . . . .	82
<b>6</b>	<b>Intelligent Hierarchical IDS for VANET</b>	<b>85</b>
6.1	Architecture of the Intelligent Hierarchical IDS . . . . .	85

6.2	Interactions and Secure Message Exchange . . . . .	89
6.2.1	Upstream Communication . . . . .	89
6.2.2	Downstream Communication . . . . .	92
6.3	Detection Algorithms and Roles . . . . .	94
6.3.1	L0 Detection . . . . .	94
6.3.2	L1 Detection . . . . .	95
6.3.3	L2 Detection . . . . .	95
6.3.4	L3 Detection . . . . .	96
<b>7</b>	<b>Implementation of VANET applications and security mechanisms</b>	<b>98</b>
7.1	Hierarchical Intelligent IDS Architecture an Application Use-Case . . . . .	99
7.2	Implementation of VPKIbrID Security Model . . . . .	100
7.2.1	Certificate and Pseudonym Certificate Authority . . . . .	101
7.2.2	Identity Manager . . . . .	102
7.2.3	Trusted Authority . . . . .	103
7.2.4	VPKIbrID Message Encryption . . . . .	104
7.3	VPKIbrID Secured Platooning Implementation . . . . .	105
7.3.1	Platooning . . . . .	105
7.3.2	Secure Platooning Implementation . . . . .	107
7.4	Agnostic Middleware for VANETs . . . . .	108
7.5	Implementation of CAM Generation Application for Message Collection . . . . .	110
7.5.1	Denial of Service Attack . . . . .	111
7.5.2	Fabrication Attack . . . . .	112
7.6	Implementation of Machine Learning Algorithms . . . . .	112
7.6.1	Implementation of ML Algorithms for Dataset Evaluation . . . . .	113
7.6.2	Hierarchical Intelligent IDS ML Algorithms Implementation . . . . .	116
7.7	Real World Platooning Implementation . . . . .	119
<b>8</b>	<b>Test and Evaluation Procedures</b>	<b>122</b>

8.1	Evaluation of Secure Communications	122
8.2	Evaluating the Impact of Biased Datasets	126
8.3	Evaluation of the Collected VANET Datasets	127
8.3.1	Evaluation of the Collected Datasets	127
8.3.2	Datasets Evaluation in the Context of an Intrusion Detection System	130
8.4	Hierarchical IDS Results Evaluation	138
8.4.1	Performance Evaluation	139
8.4.2	Ensemble based Evaluation	140
8.4.3	Rule-based Evaluation	143
<b>9</b>	<b>Conclusions and Future Work</b>	<b>144</b>
9.1	Conclusions	144
9.2	Limitation and Future Work	147

# List of Tables

2.1	VANET Attacks Classification . . . . .	26
2.2	Solution for VANET Attacks . . . . .	28
2.3	VANET Security Mechanisms Challenges . . . . .	29
3.1	Search Terms, Alternatives and Synonyms Used in the SLR . . . . .	42
3.2	Groups of the Search Terms Used in the SLR Automated Tool . . . . .	43
3.3	Data Extracted from Studies Found in the SLR . . . . .	55
3.4	Data Extracted from Studies Follow-Up Research . . . . .	58
3.5	Most Common Tools Used in Intelligent IDS Solutions Found in the SLR . . . . .	58
5.1	Road Length, Area and Maps Used for Message Collection . . . . .	79
5.2	Vehicle Density in each Geographical Map Used for Message Collection . . . . .	81
5.3	Radio Configuration of the Vehicles Used in Message Collection . . . . .	81
5.4	CAM Parameters Used in the Message Generation Application . . . . .	81
5.5	Codes of the Different Message Types Collected . . . . .	83
5.6	Parameters of the Collected Messages . . . . .	84
8.1	Comparison Between VPKIbrID Modes - Encryption Performance . . . . .	124
8.2	Comparison Between VPKIbrID Modes - Decryption Performance . . . . .	124
8.3	Comparison Between VPKIbrID Modes - Message Size (Bytes) . . . . .	125
8.4	Collected DoS Dataset Results - Interval 1 - 10% . . . . .	128
8.5	Collected DoS Dataset Results - Interval 10 - 20% . . . . .	128
8.6	Collected DoS Dataset Results - Interval 20 - 30% . . . . .	128

8.7	Collected Dataset Results - Fake Heading Attack . . . . .	129
8.8	Collected Dataset Results - Fake Speed Attack . . . . .	129
8.9	Collected Dataset Results - Fake Acceleration Attack . . . . .	129
8.10	Contents of the test and training datasets per message type . . . . .	130
8.11	ML Algorithm Configuration Parameters for Evaluation in the Context of an IDS . . . . .	131
8.12	Dataset Contents - Singular maps . . . . .	132
8.13	Dataset Evaluation - Two Dataset Grouping . . . . .	135
8.14	Cluster Division Using the Entities from the Dataset . . . . .	136
8.15	Dataset Evaluation Result per Cluster Level . . . . .	138
8.16	Number of Messages per Cluster Level . . . . .	138
8.17	Evaluation Results of the ML algorithms . . . . .	139
8.18	Comparison of the Size and Elapsed Time Taken During Training and Testing . . . . .	140
8.19	Configuration of the Algorithms for the Ensemble Learning . . . . .	141
8.20	Classification Results Using Ensemble Learning . . . . .	142
8.21	In-depth Evaluation of the Decision Stump Algorithm . . . . .	143

# List of Figures

2.1	Cryptography Basic Communication Model . . . . .	12
2.2	Symmetric Cipher . . . . .	14
2.3	Asymmetric Cipher . . . . .	14
2.4	VANET Architecture . . . . .	18
2.5	WAVE Architecture . . . . .	20
2.6	CALM Architecture . . . . .	21
2.7	ETSI ITS-G5 Architecture . . . . .	22
2.8	DSRC Protocol Stack . . . . .	23
2.9	DSRC Channels in USA . . . . .	23
2.10	DSRC Channels in Europe . . . . .	24
2.11	Platooning of Vehicles . . . . .	35
3.1	SLR Methodology . . . . .	41
3.2	Number of Studies on Intelligent IDS Published Per Year . . . . .	45
3.3	Types of Network Simulators Used in Intelligent IDS Related Works . . . . .	45
3.4	Types of Traffic Simulators Used in Intelligent IDS Related Works . . . . .	46
3.5	Origin of the Datasets Used in Intelligent IDS Related Works . . . . .	46
3.6	Types of Machine Learning Algorithms in Intelligent IDS Related Works . . . . .	47
4.1	VPKIbrID Security Model Interactions . . . . .	61
4.2	VPKIbrID-PM Format in ASN.1 . . . . .	62
4.3	VPKIbrID-EM Format in ASN.1 . . . . .	62

4.4	VPKIbrID-PKI and VPKIbrID-ABE Block Diagram . . . . .	63
4.5	VPKIbrID-PKI Encryption and Decryption Sequence Diagram . . . . .	64
4.6	VPKIbrID-ABE Encryption and Decryption Sequence Diagram . . . . .	65
4.7	Request VPKIbrID Token From the IdM . . . . .	67
4.8	Request VPKIbrID PCs From the PCA . . . . .	68
4.9	Request VPKIbrID ABE Keys From the TA . . . . .	69
4.10	VPKIbrID Obtaining User Token . . . . .	70
4.11	Using the VPKIbrID User Token to Access the Vehicle . . . . .	71
4.12	VPKIbrID Security Model Interactions with Administrator . . . . .	73
6.1	Hierarchical Intelligent IDS Security Framework Functional Architecture . . . . .	86
6.2	Hierarchical Intelligent IDS Architecture . . . . .	88
6.3	Upstream Communication and Processing of the Received CAMs . . . . .	91
6.4	Downstream Communication of the Produced Rules and Models . . . . .	93
6.5	Cluster Level Needs vs. Characteristics . . . . .	94
7.1	Intelligent Hierarchical IDS Architecture - Use Case . . . . .	99
7.2	VPKIbrID CA Example Configuration File . . . . .	101
7.3	VPKIbrID IdM Example Configuration Cile . . . . .	102
7.4	VPKIbrID IdM Token Example . . . . .	103
7.5	VPKIbrID Library Example Configuration File . . . . .	104
7.6	Platooning Example Configuration File . . . . .	106
7.7	Platooning Application State Transition . . . . .	107
7.8	Agnostic and Modular Architecture for the Development of Cooperative ITS Applications . . .	109
7.9	Attacker Selection Flow for Message Generation . . . . .	110
7.10	Selection of DoS Time Interval and Duration for Message Generation . . . . .	111
7.11	Web-based GUI for ML Evaluation - Processing . . . . .	114
7.12	Web-based GUI for ML Evaluation - Processing . . . . .	115
7.13	Loading Dataset Using Weka Java Library . . . . .	116



7.14	Setting Classifier Index Using Weka Java Library . . . . .	116
7.15	Adding Filter to the Classifier Using Weka Java Library . . . . .	117
7.16	Building Model With RF Using Weka Java Library . . . . .	117
7.17	Classifying Test Instances Using Weka Java Library . . . . .	117
7.18	Building Ensemble Voting Using Weka Java Library . . . . .	118
7.19	Building Ensemble Voting Using Weka Java Library . . . . .	118
7.20	Platooning Real-World Implementation Circuit . . . . .	119
7.21	Platooning Real-World Implementation Setup . . . . .	120
7.22	Platooning Real-World Implementation Application Example and Scenario . . . . .	121
8.1	Evaluation of the Datasets Produced in each Geographical Map . . . . .	133
8.2	Number of Attackers (Left-Hand Side) and Messages (Right-Hand Side) per Dataset . . . . .	133
8.3	Evaluation Using 2, 3, 4 and 5 Datasets Grouping - Random Grouping . . . . .	134
8.4	Dataset Evaluation Result per Cluster Level . . . . .	137
8.5	Custom Stacking Algorithm. . . . .	142

# Acronyms

<b>ABE</b>	Attribute-Based Encryption . . . . .	145
<b>AES</b>	Advanced Encryption Standard . . . . .	20
<b>ANN</b>	Artificial Neural Networks . . . . .	33
<b>AODV</b>	Ad hoc On-Demand Distance Vector . . . . .	51
<b>ASN.1</b>	Abstract Syntax Notation One . . . . .	61
<b>BPNN</b>	Back Propagation Neural Network . . . . .	51
<b>C-ITS</b>	Cooperative Intelligent Transportation Systems . . . . .	20
<b>CA</b>	Certification Authority . . . . .	15
<b>CALM</b>	Communications Access for Land Mobiles . . . . .	19
<b>CAM</b>	Context Awareness Message . . . . .	29
<b>CAN</b>	Controller Area Network . . . . .	119
<b>CBC-MAC</b>	Cipher Block Message Authentication Code . . . . .	20
<b>CBR</b>	Constant Bit Rate . . . . .	48
<b>CCH</b>	Control Chanel . . . . .	23
<b>CCM</b>	Counter with CBC-MAC . . . . .	20
<b>CMS</b>	Cryptographic Message Syntax . . . . .	20
<b>CN</b>	Common Name . . . . .	101
<b>CP-ABE</b>	Cipher-Policy Attribute-Based Encryption . . . . .	104
<b>CPS</b>	Cyber-Physical System . . . . .	32
<b>CPU</b>	Central Processing Unit . . . . .	145
<b>CRL</b>	Certificate Revocation List . . . . .	8
<b>DCAITI</b>	Daimler Center for Automotive IT Innovations . . . . .	80
<b>DENM</b>	Decentralized Environmental Notification Message . . . . .	29
<b>DoS</b>	Denial of Service . . . . .	145
<b>DR</b>	Detection Rate . . . . .	32
<b>DS</b>	Decision Stump . . . . .	146
<b>DSRC</b>	Dedicated Short-Range Communications . . . . .	22
<b>DTLS</b>	Datagram Transport Layer Security . . . . .	20

<b>ECC</b>	Elliptic Curve Cryptography . . . . .	19
<b>ECDSA</b>	Elliptic Curve Digital Signature . . . . .	19
<b>ECIES</b>	Elliptic Curve Integrated Encryption Scheme . . . . .	20
<b>ETSI</b>	European Telecommunications Standards Institute . . . . .	19
<b>FFNN</b>	Feed-Forward Neural Network . . . . .	50
<b>FNR</b>	False Negative Rate . . . . .	32
<b>FPR</b>	False Positive Rate . . . . .	32
<b>GPS</b>	Global Positioning System . . . . .	27
<b>HALL</b>	High Availability and Low Latency . . . . .	23
<b>HGNG</b>	Hierarchical Growing Gas Network . . . . .	53
<b>HTTP</b>	HyperText Transfer Protocol . . . . .	17
<b>HTTPS</b>	HyperText Transfer Protocol Secure . . . . .	24
<b>I2V</b>	Infrastructure to Vehicle . . . . .	17
<b>IdM</b>	Identity Manager . . . . .	4
<b>IdP</b>	Identity Provider . . . . .	16
<b>IDS</b>	Intrusion Detection System . . . . .	144
<b>IEEE</b>	Institute of Electrical and Electronics Engineers . . . . .	29
<b>I-GHSOM</b>	Improved Growing Hierarchical Self-Organized Map . . . . .	53
<b>IoT</b>	Internet of Things . . . . .	59
<b>IP</b>	Internet Protocol . . . . .	29
<b>IPSec</b>	Internet Protocol Security Protocol . . . . .	20
<b>ISO</b>	International Organization for Standardization . . . . .	20
<b>ITS</b>	Intelligent Transportation Systems . . . . .	1
<b>ITS-LCI</b>	ITS Local Communication Interface . . . . .	108
<b>JKS</b>	Java Key Store . . . . .	102
<b>JPBC</b>	Java Pairing-Based Cryptography . . . . .	104
<b>JSON</b>	JavaScript Object Notation . . . . .	102
<b>JWE</b>	JSON Web Encryption . . . . .	17
<b>JWS</b>	JSON Web Signature . . . . .	17
<b>JWT</b>	JSON Web Token . . . . .	17
<b>LA</b>	Learning Automata . . . . .	47
<b>LLC</b>	Logical Link Control . . . . .	23
<b>LMT</b>	Logistic Model Tree . . . . .	146
<b>LTC</b>	Long-Term Certificates . . . . .	65
<b>MAC</b>	Message Authentication Code . . . . .	63
<b>MAE</b>	Mean Absolute Error . . . . .	139

<b>MANET</b>	Mobile Ad hoc Network . . . . .	17
<b>MIB</b>	Management Information Bases . . . . .	109
<b>MCC</b>	Matthews Correlation Coefficient . . . . .	49
<b>ML</b>	Machine Learning . . . . .	144
<b>MLP</b>	Multilayer Perceptron . . . . .	146
<b>MOVE</b>	MObility VEhicles . . . . .	45
<b>MPR</b>	MultiPoint Relay . . . . .	51
<b>NIST</b>	National Institute of Standards Technology . . . . .	19
<b>NN</b>	Neural Networks . . . . .	144
<b>ns-2</b>	Network Simulator 2 . . . . .	144
<b>ns-3</b>	Network Simulator 3 . . . . .	144
<b>OBU</b>	On-Board Unit . . . . .	1
<b>OFDM</b>	Orthogonal Frequency Division Multiplexing . . . . .	24
<b>OSI</b>	Open Systems Interconnection . . . . .	22
<b>OSM</b>	Open Street Maps . . . . .	78
<b>OU</b>	Organizational Unit . . . . .	101
<b>PC</b>	Pseudonym Certificate . . . . .	30
<b>PCA</b>	Pseudonym CA . . . . .	30
<b>PKI</b>	Public Key Infrastructure . . . . .	145
<b>PMP</b>	Platoon Management Protocol . . . . .	36
<b>POS</b>	Proportional Overlapping Scores . . . . .	50
<b>QP</b>	Quadratic Programming . . . . .	34
<b>RF</b>	Random Forest . . . . .	146
<b>RQ</b>	Research Question . . . . .	40
<b>RMSE</b>	Root Mean Square Error . . . . .	139
<b>RSU</b>	Road Side Unit . . . . .	145
<b>SAE</b>	Society of Automotive Engineers . . . . .	24
<b>SAML</b>	Security Assertion Markup Language . . . . .	16
<b>SAT</b>	Situation-Aware Trust . . . . .	31
<b>SOAP</b>	Simple Object Access Protocol . . . . .	17
<b>SQ</b>	Search Query . . . . .	42
<b>SCH</b>	Service Channel . . . . .	23
<b>SLR</b>	Systematic Literature Review . . . . .	144
<b>SMO</b>	Sequential Minimal Optimization . . . . .	34
<b>SP</b>	Service Provider . . . . .	16
<b>SUMO</b>	Simulation of Urban Mobility . . . . .	144

<b>SVM</b>	Support Vector Machine . . . . .	33
<b>WAVE</b>	Wireless Access in Vehicular Environment . . . . .	19
<b>WLAN</b>	Wireless Local Area Network . . . . .	32
<b>WSN</b>	Wireless Sensor Network . . . . .	32
<b>TA</b>	Trusted Authority . . . . .	16
<b>TCP</b>	Transport Control Protocol . . . . .	109
<b>TLS</b>	Transport Layer Security . . . . .	37
<b>TPR</b>	True Positive Rate . . . . .	52
<b>TRACI</b>	Traffic Control Interface . . . . .	105
<b>UDP</b>	User Datagram Protocol . . . . .	20
<b>VANET</b>	Vehicular Ad hoc Network . . . . .	144
<b>V2I</b>	Vehicle to Infrastructure . . . . .	17
<b>V2V</b>	Vehicle to Vehicle . . . . .	17
<b>V2X</b>	Vehicle to Anything . . . . .	21
<b>VA</b>	Validation Authority . . . . .	28
<b>VPKIbrID</b>	Vehicular Ad hoc Network Public Key Infrastructure and Attribute-Based Encryption with Identity Manager Hybrid . . . . .	145
<b>VPKIbrID-ABE</b>	VPKIbrid Attribute Base Encryption . . . . .	145
<b>VPKIbrID-EM</b>	VPKIbrid Encrypted Message . . . . .	61
<b>VPKIbrID-PKI</b>	VPKIbrid Public Key Infrastructure . . . . .	145
<b>VPKIbrID-PM</b>	VPKIbrid Plain Message . . . . .	61
<b>VSimRTI</b>	V2X Simulation Runtime Infrastructure . . . . .	80
<b>XML</b>	Extensible Markup Language . . . . .	17

# Chapter 1

## Introduction

Vehicular Ad hoc Networks ([VANETs](#)) are the core of Intelligent Transportation Systems ([ITS](#)), allowing vehicles to communicate among themselves and with the infrastructural network. They enable the implementation of multiple types of applications with the primary goal of improving road safety by reducing accidents and traffic congestion. Nevertheless, these are complex networks with volatile architectures and ever-changing members with an inherently insecure communication medium (the air), making them particularly attractive to attackers, where even small attacks may cause severe damage.

On security concerns, [VANETs](#) involve multiple entities that pose important security challenges. These are the vehicles, their On-Board Units ([OBUs](#)), the drivers and the infrastructural Road Side Units ([RSUs](#)), and third-party service suppliers (connectivity or content). Hence, the implementation of security measures being of significant importance.

Several studies have been done on preventing [VANET](#) attacks, generally by using cryptographic tools [[1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#)], which aim to prevent and/or mitigate existing attacks. However, these methods cannot detect or prevent all attacks. Some examples of attacks that traditional methods cannot prevent are [[7](#)] Denial of Service ([DoS](#)), Black Hole, Grey Hole, and Sybil attack. The usage of cryptography can even increase the possibility of [DoS](#)[[8](#)]. If the messages are signed, checking for the signatures on fake ones can overload an entity.

One possible solution for this problem is the usage of an Intrusion Detection System ([IDS](#)). These can detect attacks and trigger a response, minimizing the effects on the targeted system [[9](#)]. Traditional [IDSs](#) assume that the behavior of an intruder will be noticeable from normal network operation. Signature detection systems compare the behaviors of the attackers with known attacks, assuming no new attacks will happen. Anomaly detection systems focus on detecting significant deviations from normal behavior. However, the definition of these criteria is difficult to attain[[10](#)]. The usage of intelligent algorithms enables the [IDS](#) to learn previous attacks and discover new ones.

This research proposal is on the identification, classification, and characterization of the most critical security threats in VANET. It should combine highly dynamic contextual vehicular information with central knowledge built from previously collected data (either from previous security analysis, derived from collected data, or threats to services or identities). This information should enable the design and validation of a new type of IDS by using distributed and intelligent algorithms structured in a hierarchical architecture with multiple clusters, attributing different roles and detection types to each level, according to its characteristics, capabilities, and needs. This new type of IDS should enable each network node to identify the attacks mentioned above and, thus, protect the VANET and its vehicles without compromising vehicles or associated services.

This thesis proposal is aligned with the project Easy Ride: Experience is Everything - 039334, an R&D project on VANETs, resulting from the University of Minho and Bosch Car Multimedia's collaboration, in which the Computer Communication Group participates. Within this project framework, it is expected to identify scenarios, use cases, and datasets useful for the thesis work.

## **1.1 Objectives**

This work focus on the definition, evaluation, and test of a new IDS targeted at the context of VANETs. It should take advantage of intelligent and distributed algorithms to improve the IDS dynamic adaptation, performance, and detection capacity. Moreover, the IDS design follows a hierarchical architecture composed of multiple levels, grouping nodes of similar characteristics, capabilities, and needs. Thus, each level may use a specific detection type that is better suited for its requirements. The goals of this research are the following:

**Survey VANET characteristics, attacks, and attackers -** VANETs are the underlying communication framework used by the entities composing the architecture to be designed. Thus, information on VANET characteristics, security requirements, attacks and attackers, and secure communications models should be collected. The survey should provide a strong enough basis for the rest of the work.

**Survey IDSs in the VANET context -** IDSs have diverse designs and characteristics which can be better suited for different scenarios. These can also be deployed in different network locations and entities. Thus a survey gauging the suitability of this type of solution in this context is needed. The survey should also gather information on the more common approaches, simulation frameworks, and datasets.

**Research and implement a secure communication framework -** Because the proposed IDS design is divided into multiple clusters, their entities need a channel enabling the secure exchange of information. Thus,

existing security frameworks for VANETs should be evaluated. If they do not satisfy the security requirements, a new one should be designed. The chosen security model should be implemented and tested. It will serve as a basis for the IDS entities to communicate securely. Also, due to the needs of the IDS and the intrinsic nature of VANET communications, the security framework should consider the demand for broadcast communications.

**Collection of VANET datasets with attacks -** Machine Learning (ML)-based IDSs need large sets of data to be trained and tested. So VANET datasets should be surveyed. If none is found, these should be collected by following a well-specified methodology containing both normal and attack messages. Thus VANET attacks, with particular attention to the ones unpreventable by cryptographic tools, should be identified and reproduced. Attacks and normal messages should be collected in a significant number and relevance to train and test the prototype.

**Evaluation of ML algorithms for attack detection in the VANET context -** The IDS proposed in this work is aided by intelligent algorithms, aiming to enhance its detection capabilities. These types of algorithms provide flexibility and good anomaly detection capabilities. So, it is also a goal to research intelligent algorithms, mainly in the context of attack detection. The ML algorithms should be trained and tested, simulating their location in different network locations. Thus, the ML algorithm at each level should only have access to the limited data collected by its nodes, providing insight on the best combination of algorithms for each network level.

**Design the IDS architecture -** The knowledge obtained from the previous goals should enable the design of the IDS. It follows a hierarchical model, grouping entities with similar characteristics in different network levels. Thus, enabling the implementation of different functionalities and detection types according to each level's entities' capabilities and needs. Then, the architecture as a whole can be designed. This design should contain the methodology used to maintain secure communications between entities, the ML algorithms used at each level, and the roles for each node.

**Develop a prototype implementation -** A prototype implementation of the IDS should be done as a proof of concept. It should be independent of any specific system for its easy adaptation at each VANET node. The IDS should be validated, deployed, and tested in a simulation tool and, if possible, making a real-world deployment. Therefore, a survey of simulation tools should be done to find the better suited for this implementation.

**Test the new methodology -** The final goal is to derive, validate, and test the new methodology and classify the implemented IDS regarding its characteristics, such as false negatives, false positives, precision



and accuracy in attack detection. Also, evaluating possible delays and overhead introduced by the underlying security model, and the IDS detection time should be considered.

## 1.2 Methodology of the Research

This Section presents the detailed methodology followed to fulfill the previously established objectives. This thesis should culminate in the proposal, test, and evaluation of a security architecture composed of multiple IDS (one by each VANET node) organized into a hierarchy that is able to identify attacks correctly. The nodes should be able to communicate securely, sharing attack information to enable a quicker and cooperative attack detection. The lower nodes should also send the information received to a higher level IDS for further analysis. It will foster a wider knowledge of the network and enable a more profound analysis of the network (slower), enabling it to detect other attacks undetected by the remaining nodes. The IDSs can be divided into clusters of, for example, different brands or commercial fleets to detect more particular attacks.

**Survey VANET characteristics, attacks, and attackers -** The first step is to thoroughly research the VANET related literature to survey their characteristics, security requirements, and most known attacks. The study should emphasize in attacks unpreventable through the usage of traditional security measures (cryptography-based).

**Survey IDSs in the VANET context -** The survey should have an accent on IDSs for VANETs supported by Intelligent Algorithms. This survey is supported by a Systematic Literature Review (SLR), one of the best ways to systematically cover the existing literature [11]. Additionally, the review should provide insight into the types and sources of the datasets used in other works. One of the requirements of using ML is the need for a large set of data to train and test the algorithms. If none publicly available are found, these should be synthesized, including multiple attacks and normal messages. The IDS designs found in the literature should be evaluated to find the best suited for this scenario. This step also consists of the research and identification of the simulation framework to be used.

**Research and implement a secure communication framework -** During the curricular MAP-i plan, the unit "free option" involves research work that can be done in the thesis's scope. The work done in this curricular unit was the evaluation, definition, and implementation of a security model for VANETs. Firstly, the existing models were studied, surveying their features and drawbacks. The implemented security model uses a Public Key Infrastructure (PKI) base improved with Attribute-Based Encryption (ABE) and an Identity Manager (IdM) to provide privacy protection. This work has resulted in a scientific paper and has been published in an

international conference.

**Collection of VANET datasets with attacks -** Carefully research the literature on IDS for VANET, focusing on ML-based research work. The research should provide insight on the data source used for the datasets used by others or, if these are not available, the methodology used to generate the needed data. If no publicly available data source is found, the datasets are to be synthesized using simulation to generate the attacks and normal messages, which will be collected to create datasets.

**Evaluation of ML algorithms for attack detection in the VANET context -** There are multiple types of ML algorithms with different characteristics. These should be studied in the context of the hierarchical IDS aimed at VANETs. Thus, each level has nodes with different requirements that may need algorithms with specific characteristics. The algorithms should be trained and tested using the previously collected messages. These algorithms should be evaluated to find the best-suited that can balance detection rate, false positives, and false negatives, in conjunction with detection speed and complexity according to the needs of each each.

**Design the IDS architecture -** The main goal of this work is the research of a new hierarchical IDS. The IDS should be divided into multiple levels, grouping entities with the same characteristics and needs. Thus, the most Central Processing Unit (CPU)-powerful entities can perform the more complex operations, such as, creating ML models and analyzing all the data, while the entities with less CPU power can use the rules and models created to perform detection. Moreover, the architecture should include the algorithms each node should use to be more efficient, the underlying security model to create a secure channel for all the entities to communicate and their interactions.

**Develop a prototype implementation -** The developed prototype should include tools that enable secure communication between the entities. So, in addition to implementing the ML algorithms, the security model and all the architecture components should be implemented, enabling the evaluation of the proposed methodology at all levels.

**Test the new methodology -** The real-world implementation should be done in the context of the research project, where this work is being developed. It is a partnership between the University of Minho and an external enterprise that may enable real-world application testing.

## 1.3 Scientific Contributions

The research work performed during this Ph.D. thesis resulted in multiple scientific publications. Moreover, this thesis work was aligned with CAR2X, an R&D project on [VANETs](#), resulting from the collaboration of the University of Minho and Bosch Car Multimedia, which enabled the deployment of a platooning application in real-world vehicles in a laboratory environment.

**Simulation and Testing of a Platooning Management Protocol Implementation [12]** Bruno Ribeiro, Fábio Gonçalves, Alexandre Santos, Maria João Nicolau, Bruno Dias, Joaquim Macedo and António Costa, 15th IFIP WG 6.2 International Conference on Wired/Wireless Internet Communications (WWIC 2017), St. Petersburg, Russia, June 21-23, 2017.

**A New Approach on Communications Architectures for Intelligent Transportation Systems [13]** Susana Sousa, Alexandre Santos, António Costa, Bruno Dias, Bruno Ribeiro, Fábio Gonçalves, Joaquim Macedo, Maria João Nicolau and Oscar Gama, 12th International Conference on Future Networks and Communication (FNC-2017), Leuven, Belgium, July 24-26, 2017.

**Hybrid Model for Secure Communications and Identity Management in Vehicular Ad Hoc Networks [14]** Fábio Gonçalves, Alexandre Santos, António Costa, Bruno Dias, B. Ribeiro, J. Macedo, M. J. Nicolau, S. Sousa, O. Gama, S. Barros, V. Hapanchak, 9th International Congress on Ultra Modern Telecommunications and Control Systems (ICUMT'2017), Munich, Germany, November 6-9, 2017.

**PlaSA - Platooning Service Architecture [15]** Bruno Ribeiro, F. Gonçalves, V. Hapanchak, O. Gama, S. Barros, P. Araujo, António Costa, M. João Nicolau, , Bruno Dias, Joaquim Macedo, Alexandre Santos, 8th ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (MSWIM'2018), Montreal, Canada, October 28 - November 2, 2018.

**Secure Management of Autonomous Vehicle Platooning [16]** Fábio Gonçalves, B. Ribeiro, V. Hapanchak, S. Barros, O. Gama, P. Araujo, M. João Nicolau, Bruno Dias, Joaquim Macedo, António Costa, Alexandre Santos, 14th ACM Symposium on QoS and Security for Wireless and Mobile Networks (MSWIM'2018), Montreal, Canada, October 28 - November 2, 2018.

**Agnostic and Modular Architecture for the Development of Cooperative ITS Applications [17]** Bruno Dias, Alexandre Santos, António Costa, B. Ribeiro, F. Gonçalves, Joaquim Macedo, M. João Nicolau, O.

Gama, S. Sousa, Journal of Communication Software and Systems, September, 2018.

**Simulation and Testing of a Platooning Cooperative Longitudinal Controller [18]** Vadym Hapanchak, António Costa, Joaquim Macedo, Bruno Dias, M. João Nicolau, Alexandre Santos, B. Ribeiro, F. Gonçalves, O. Gama, P. Araujo, Guimarães, Lecture Notes of the Institute for Computer sciences, Social-Informatics, and Telecommunications Engineering (LNICST), Portugal, November 21-23, 2018.

**A Systematic Review on Intelligent Intrusion Detection systems for VANETs [19]** Fábio Gonçalves, B. Ribeiro, O. Gama, Bruno Dias, Joaquim Macedo, M. João Nicolau, António Costa, Alexandre Santos, 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT'2019), Dublin Ireland, October 28-30, 2019

**Synthesizing Datasets with Security Threats for Vehicular Ad-Hoc Networks [20]** Fábio Gonçalves, B. Ribeiro, O. Gama, J. Santos, António Costa, Bruno Dias, M. João Nicolau, Joaquim Macedo and Alexandre Santos, 2020 IEEE Global Communications Conference (GLOBECOM'2020), Taipei, Taiwan, December 7-11, 2020

**Evaluation of VANET Datasets in context of an Intrusion Detection System [21]** Fábio Gonçalves, Joaquim Macedo, Alexandre Santos, 29th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2021), Hvar, Croatia, September 23-25, 2021

**Intelligent Hierarchical Intrusion Detection System for VANETs [22]** Fábio Gonçalves, Joaquim Macedo, Alexandre Santos, 13th International Congress on Ultra Modern Communications and Control Systems (ICUMT), Online, October 25-27, 2021

**An Intelligent Hierarchical Security Framework for VANETs [23]** Fábio Gonçalves, Joaquim Macedo, Alexandre Santos, Information 12, 2021

## **1.4 Organization of the Document**

The present document starts with the introduction (Section 1), which describes the problem tackled in this Ph.D. and the proposed solution. Then in Section 1.1 are presented the research goals of this work and in Section 1.2 the methodology that will be followed to achieve them successfully. The research work done

resulted in multiple published scientific papers, which are described in Section 1.3. This Chapter ends with this Section, which describes the organization of this document.

Chapter 2 presents the technologies related to this Ph.D. thesis work. Firstly, the security communication basics are introduced, including the fundamentals of cryptography, PKIs, Certificate Revocation Lists (CRLs), ABE, and IdM. Then, are presented the VANETs and their characteristics, primary standards, and most common architectures. This research focuses on security, so Sections 2.3, 2.4, 2.5 focus on this VANET area, presenting the security requirements, main attacks and their security solutions, and the existing security models for VANETs. Section 2.6 gives an overview of IDSs, presenting the different detection types, the metrics used for their evaluation, and the different types of architectures existing. This work aims to use ML algorithms in conjunction with IDSs, which are described in Section 2.7. This Section describes the types of machine learning and some of the existing algorithms. Finally, and due to its relevance to the rest of the work, the Platooning application is described in Section 2.8, which includes both an overview of the functioning of the Platooning and its security requirements.

In Chapter 3 is presented the related work. It serves as a basis for this Ph.D. work, enabling the evaluation of the implementability and feasibility of an Intelligent IDS for VANETs. The related work was surveyed through a thorough SLR that was performed following a well-defined methodology. Thus, enabling the identification of relevant research work while decreasing the probability of biased results. This review provided insight into the most common approaches in this area, such as IDS architectures, ML algorithms, simulators, and used datasets.

The architecture has multiple entities that need to communicate securely over the insecure environment, that is the VANET, to cooperate with each other. Chapter 4 presents the security model designed with that purpose called Vehicular Ad hoc Network Public Key Infrastructure and Attribute-Based Encryption with Identity Manager Hybrid (VPKIbrID), including its different encryption modes and messages, and the interactions between the entities within the VPKIbrID model. Then, in Section 4.4, it is described how the VPKIbrID model can be applied to a specific situation, in this case, the platooning application.

The IDS proposed takes advantage of the capabilities provided by ML algorithms to improve its detection. These types of algorithms need large sets of data for their training and testing. However, as indicated in Chapter 3, no publicly datasets were found in the SLR, as most research works do not publish their datasets. So, in Chapter 5, it is presented the methodology used to create the datasets in this work. Firstly in Section 5.1, are presented the scenarios used for the message collection and indicated the type of application used for the traffic generation. Then in Section 5.2, the geographical maps, their characteristics, and Simulation of Urban Mobility (SUMO) counterparts are shown. Multiple datasets were used to introduce randomness in the data collected, enabling the detection of different attacks in different situations. Section 5.3 describes how the simulation was performed and which types of attacks were used. Lastly, it is described how the data was collected and its format in Section 5.4.

Then, Chapter 6 presents the designed *Intelligente Hierarchical IDS*. First, in Section 6.1, the design of the architecture is presented. It shows the functional division of the architecture that joins entities with similar characteristics and functionalities into clusters at the same level. And, furthermore, their interactions, message exchange, and underlying security framework. In Section 6.2, the complete security interactions are described. At this level, the security is represented through a secure channel that enables all entities to communicate securely over an insecure channel. Section 6.3 describes the detection at each level, including the *ML* algorithm used and the functionality of each cluster.

As a proof of concept and to provide results for evaluating the proposed technologies, these were implemented as described in Chapter 7. Section 7.1 shows the *Intelligent Hierarchical IDS* applied to a specific use case. In this case, it describes an implementation that groups the vehicles into platooning, using the already established cluster mechanisms and communications. Section 7.2 presents the implementation of the security model *VPKIbrID*. This model comprises multiple entities responsible for the generation of different cryptographic material needed for secure interactions. A simple implementation of those entities was made, including an easy-to-use library that provides a simple way to add *VPKIbrID* to any application. In Section 7.3, the *VPKIbrID* security model is used to secure communications in a Platooning. The implementation of the agnostic architecture used to provide seamless access to the vehicle internals and communication modules is presented in Section 7.4. Section 7.5 describes the implementation of the application used to generate the messages collected to build the datasets indicated in Chapter 5, including the simulation tools and the methodology used to create the multiple simulated attacks. In Section 7.6 is described the different implementations of the *ML* algorithms used. Finally, in Section 7.7 is described a real-world implementation in a controlled environment of a Platooning application using real vehicles and physical communication devices.

The results obtained from the applications implemented in Chapter 7 are then presented in Chapter 8. The results allow the evaluation of the technologies presented and their comparison with existing work. Section 8.1 presents the results obtained for the *VPKIbrID* model implementation, evaluating the multiple encryption modes and providing insight into the impact of the cryptography schemes used. The secure platooning implementation did not produce any results due to the characteristics of the simulator, as it does not consider the time taken during the cryptographic operations. Section 8.2 describes a test made through a simple application that generates messages with a constant frequency. The goal of this application was to test the influence of having a biased dataset, demonstrating its impact on the results. Then, in Section 8.3, two different results are presented; in Section 8.3.1, the contents of the collected datasets are presented and how each parameter varies in the normal messages and attacks; in Section 8.3.2, are presented the results obtained from the evaluation done by feeding the produced datasets to an *ML* algorithm. The goal of the evaluation is to gauge the impact of having datasets from multiple sources and using datasets from the point of view of different network locations. Section 8.4 presents the evaluation of multiple *ML* algorithms, optimizing the detection of attacks using the datasets collected. Furthermore, these results enable the selection of the best-suited algorithm for each level of the cluster.

Finally, the conclusions are presented in Chapter 9, describing the major results and solutions of the work performed. This Chapter terminates with the limitations presented by the proposed work, which can, in some way, introduce some bias in the obtained results and the further research work that can complement this thesis, such as real-world implementation, testing, and deeper analysis that could not be completed.

# Chapter 2

## Background

This Chapter describes the current state of the tools and technologies related to this work and makes an overview of the more general and fundamental security mechanisms.

Firstly, Section 2.1 starts by introducing the main cryptographic mechanisms and, due to their relationship with this work, [IdM](#), [PKI](#), [ABE](#), and [CRL](#). Section 2.2 makes a general description of [VANETs](#), indicating their standards, main features, and characteristics. The security requirements for [VANETs](#) are presented in Section 2.3, providing insight on the security measures needed for their entities to communicate. The most common types of [VANET](#) attackers, their motivations, and most known attacks are presented in Section 2.4, helping to understand the threats to the system and improve the architecture design. Additionally, it provides information on how to build attacks to train the [IDS](#). In Section 2.5, a summary of the security models for [VANETs](#) found in the literature is made. These are to be used as an underlying framework to provide secure communications between the architecture entities. Sections 2.6 and 2.7 are more directly related to the work, presenting the existing [IDSs](#) related works and describing some of the most known ML algorithms.

### 2.1 Security

Security is one of the most fundamental components of any communication system. Secure communication can provide multiple assurances, from privacy and anonymity to authentication and non-repudiation. Without these properties, no entities can communicate with any certainty about the parties involved, message origin, or alterations suffered by the information exchanged. Furthermore, it is not possible to have a secret exchange of information or maintain privacy. Security technologies are also the basis of identity managers such as Google or Facebook, providing secure authentication in multiple websites while providing privacy.

This Section presents a brief overview of some more general security technologies related to this work.



It starts with a short description of the most basic cryptographic tools (Section 2.1.1). Furthermore, due to their relationship with this research, PKI, CRL, ABE and, IdM are also described.

### 2.1.1 Cryptography

Cryptography is the science of secret writing. It is traditionally used to enable secret communications through an insecure channel. To do so, the sender obfuscates the message and sends it to the receiver. The message is obfuscated by a method called cipher which, transforms a plaintext message into ciphertext. The inverse process is called decipher. These are both controlled by one or several cryptographic keys [24].

The basic communication model (Figure 2.1), used to demonstrate cryptographic applications, consists of three entities: Alice, Bob, and Eve. Alice and Bob represent two entities trying to communicate securely over an insecure channel. Eve is an attacker trying to eavesdrop on the communication. It is assumed that Eve has great processing power, can see all exchanged data, and inject and modify data. Moreover, Eve knows all communication protocols and ciphers[24].

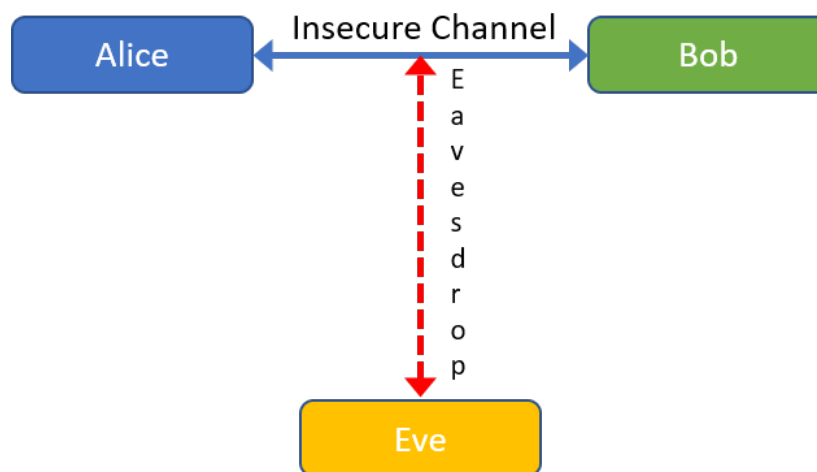


Figure 2.1: Cryptography Basic Communication Model

Securing communications over an insecure channel is the main cryptography goal. Using ciphers to obfuscate plaintexts is not enough to assure security in the communication between entities. So, a set of fundamental security objectives was defined. These should be fulfilled to provide a secure system. Although these may not apply to every situation, they are a good starting point. The fundamental security objectives are as follows:

- **Confidentiality:** Ensures that the information is kept secret to everyone except for the destination entity. Any message exchanged between Alice and Bob should not be understandable by Eve;

- **Integrity:** If any part of the message is altered in its path, it should be noticeable, assuring that the data is not tempered. Bob should be able to verify if the message sent by Alice did not suffer any alteration since its origin;
- **Origin Authentication:** Provides data authenticity verification. Bob should be able to verify the authenticity of the data received. Moreover, he should be able to verify that the data sent by Alice was really sent by Alice;
- **Entity Authentication:** The identity of an entity should be verifiable. Bob should be able to verify the identity of Alice;
- **Non-repudiation:** Prevents any entity from denying previous action or commitments. Suppose Bob receives a message from Alice, besides being able to verify that it originated from Alice, he will be able to convince third parties that it has originated from Alice. Moreover, it prevents Alice from denying sending the message.

The main security mechanisms that provide cryptographic properties essential to secure a communication channel and fulfill the fundamental objectives are the following: ciphers, hash functions, digital signatures, and digital certificates. Each provides its own specific attributes and characteristics and is typically used to provide a secure channel. These are presented next.

## **Ciphers**

Ciphers allow entities to obfuscate the information, preventing unallowed entities from accessing their content, thus keeping the information confidential. Depending on the types of used keys, ciphers can be categorized into symmetric and asymmetric. Symmetric ciphers (Figure 2.2) use the same key for ciphering and deciphering. So, entities using symmetric ciphers need to pre-exchange them using a secure channel. Symmetric systems only assure confidentiality. Asymmetric keys (Figure 2.3), on the other hand, use different keys for the cipher and decipher process. The cipher and decipher keys constitute a key pair usually denominated as "public" and "private" keys, respectively. Asymmetric systems do not need key pre-exchange, as the public key used to cipher can be distributed freely. It assures that messages ciphered with this key will only be readable by the correspondent private key owner.

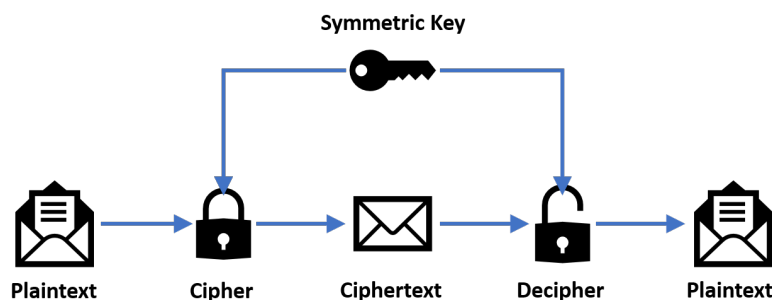


Figure 2.2: Symmetric Cipher

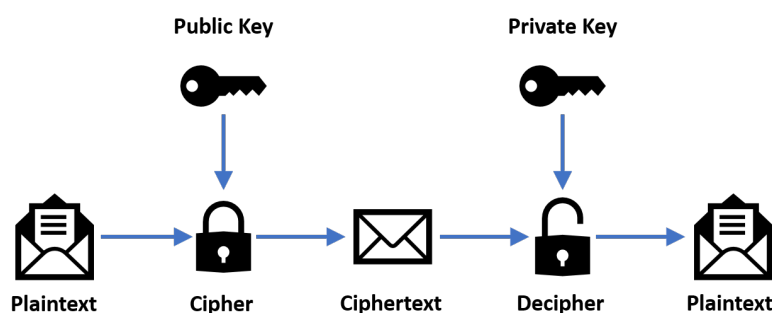


Figure 2.3: Asymmetric Cipher

## Hash Functions

Hash functions are cryptographic functions that allow the creation of a special summary from a message. A single bit changed in the original data results in a completely different hash. They are somewhat similar to checksum but purposely designed to detect deliberate changes. Moreover, they need to consider that an attacker may know the hash function calculation algorithm and will try to explore its vulnerabilities. They allow entities to easily verify if a message suffered any alteration in its path, assuring integrity [24].

## Digital Signatures

Digital signatures are a mathematical function used to sign messages. These must fulfill a set of requirements: if B receives a message (M) signed by A, B must be able to validate the signature of A in M; no entity should be able to forge A signature; A should not be able to deny signing a message (if so, it should be possible for a third party to verify it). These can be made by using symmetric or asymmetric cryptography.

## Digital Certificates

Digital certificates are digital documents that connect a public key to an entity, preventing impersonation by using, for example, a fake public key. In their simpler form, they contain a name and a public key. Digital certificates should also include the expiration date and the Certification Authority (CA) name that has emitted it. They are issued by a CA that may be any trusted central administration entity. Any third-party entity should be able to verify the identity of the certificates and the links with the issued certificate keys.

### 2.1.2 Public Key Infraestructure

PKI is a framework constituted by a group of people, processes, policies, hardware, and software [25]. It has been one of the most widely used security technologies to secure communications over the internet. The PKI provides tools to generate, manage, store, deploy and revoke public key certificates. Its goal is to create a reliable way to link an identity to a public key [25]. Thus, one of its central entities is the CA. It is an entity trusted by all entities in the system that can attest to the veracity of the certificate. To do so, it signs all the certificates it generates [25, 26].

It offers confidentiality, integrity, authenticity, and non-repudiation by recurring to the previously mentioned cryptographic tools, such as Digital Certificates and Public Key Cryptography.

### 2.1.3 Certificate Revocation Lists

The basis of PKI are certificates, which link an identity to a public key. These usually have an expiration date, indicating the date from which the certificates become invalid. During this time, as a trusted authority signs certificates, the certificates are valid and trustworthy. So, if any entity that has been issued one becomes compromised, a mechanism is needed to revoke the certificate. The revocation is usually done by distributing CRLs, which indicate the certificates that are not yet expired and digitally signed by a CA but have become invalid.

Using CRLs has, however, some drawbacks, including scalability, time constraints, and connectivity. Scalability issues are due to the sheer size of the list and the nodes it has to reach. The time constraints refer to the strenuous task needed to update the CRL constantly. Finally, the connectivity is relevant mainly to a VANET, where some nodes may not always be connected. For example, in a rural environment, the vehicle may be days without connecting with other vehicles or infrastructural networks to download the CRL.

Nonetheless, there are multiple research works in the literature that tackles the CRL distribution in the literature. Authors in Papadimitratos et al. [27] propose a scheme that relies on RSUs to distribute large CRLs across wide areas within minutes needing a low quantity of data to be exchanged. Using an epidemic

distribution of the CRL, a different strategy is proposed by authors in Laberteaux et al. [28] and Haas et al. [29]. Nowatkowski et al [30], builds on top of the epidemic strategy by using the Most Pieces Broadcast approach. It elects a single node with the most CRL pieces to broadcast, so the throughput will be the highest possible within the constraints of the channel.

### 2.1.4 Attribute Based Encryption

ABE [31] is a cryptographic method that uses attributes to cipher the data. It does not need the usage of certificates or key exchange [31]. Unlike PKI schemes, it allows data to be ciphered for multiple targets. For example, data can be ciphered to be only readable by all users who are administrators of the IT department at the University of Minho. The usage of attributes instead of public keys or identities can be very advantageous. The message can even be ciphered before having a concrete target. Moreover, the possibility of ciphering data with attributes means that the encryption process provides access control intrinsically.

It makes use of a central Trusted Authority (TA), which is the only one capable of generating private keys. This entity is also responsible for generating public parameters that are used to create encryption keys from attributes.

The main drawbacks are related to the TA and the cipher efficiency. The TA presents a single point of failure, which attackers can more easily target and, if offline, disable the communications capabilities of the affected entities. However, it may not represent a significant problem because entities only need to contact the TA to renew decryption keys. Therefore, not needing the constant support of the TA.

ABE schemes are less common and, as such, are less implemented, which may difficult their adoption. Their ciphers are usually less efficient than traditional PKI, being much slower for the same security level.

### 2.1.5 Identity Manager

IdM is a concept usually more associated with internet and web-based services [32]. It was born due to the need to grant authenticated and authorized users access to protected services and resources. It does so while implementing privacy protection mechanisms. IdMs consist of technologies and policies that represent entities and their digital identities, facilitating their management. Ferdous et al. [32] describe the entities constituting an IdM system as the Service Provider (SP) and the Identity Provider (IdP). SP is an entity that provides resources, or services, which the authenticated user wants to access. The IdP provides the digital identities of the clients, enabling the SP to provide the service.

The most used and widely accepted IdM systems are [32]: OAuth [33], OpenID Connect [34] and Security Assertion Markup Language (SAML) [34].

**SAML** is an Extensible Markup Language (**XML**) based language that defines the syntax and processing semantics of a system entity's assertions about a subject. It is used for exchanging authentication and authorization information between parties. **SAML** uses **XML** namespaces and is embedded in other structures such as HyperText Transfer Protocol (**HTTP**) POST or Simple Object Access Protocol (**SOAP**) messages. A request/response protocol is used to allow an **SP** to request identification of a particular user from the **IdP**. The last responds with the information using an assertion [32].

Oauth is an authorization framework that enables third-party applications to access a service (typically **HTTP**). It is based on token exchange to enable access right delegation. This token is opaque and has, usually, an expiration date [33].

OpenID Connect is built on top of Oauth, using the same type of interactions. In this case, instead of using an opaque token, it uses a JSON Web Token (**JWT**) [35]. It is an URL-safe way to represent claims to be transferred between entities. The **JWT** can be used to generate a JSON Web Signature (**JWS**) to provide authentication, and then it can be ciphered, creating a JSON Web Encryption (**JWE**).

## 2.2 Vehicular Ad hoc Networks

The increase of driving people leads to more traffic congestion in cities which, in turn, leads to more accidents. **ITS** is a set of applications and services that aims to facilitate transportation and make roads safer.

**VANETs** (see Figure 2.4) allow the several **ITS** nodes of the network to communicate. They are a specific type of Mobile Ad hoc Network (**MANET**) that provides communications between vehicles and roadside equipment. **VANETs**, are used by smart vehicles to communicate with each other, enabling new functionalities to be provided. Their main objective is to provide a safer road by avoiding accidents and traffic congestions while protecting the privacy of the drivers.

It is a network with a difficult control and organization, where vehicles can belong to a self-organizing network without prior screening or knowledge of each other presence [36]. There are two types of nodes: the **OBU** and the **RSU**. **RSUs** are located alongside the road and constitute the network infrastructure. The **OBUs** are installed in mobile nodes, such as vehicles [36].

Communication in **VANET** can be classified into three big groups: communication between vehicles - Vehicle to Vehicle (**V2V**); communication between the vehicles and the infrastructure - Vehicle to Infrastructure (**V2I**); and communication between the infrastructure and the vehicles - Infrastructure to Vehicle (**I2V**).

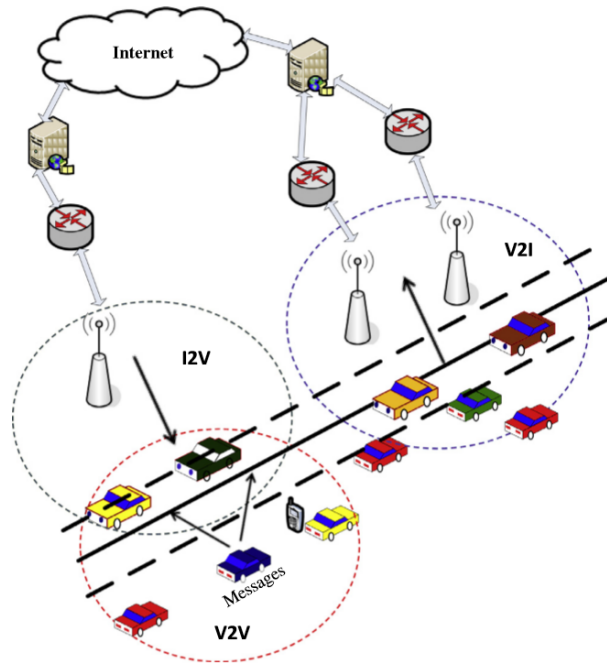


Figure 2.4: VANET Architecture (From [36])

### 2.2.1 Characteristics

Due to their nature and communication medium (the air), VANETs have some characteristics that make them especially attractive to attackers. The majority of its nodes have great mobility as they are located in vehicles. They can have different speeds and directions, making the connection between them very intermittent. According to Mejri et al. [7], the main VANET characteristics are the following: high mobility, dynamic topology, frequent disconnections, availability of the transmission medium, the anonymity of the transmission support, limited bandwidth, attenuation, limited transmission power. Moreover, these networks also have near limitless energy, storage, and computing capacity.

The availability of the transmission medium and the anonymity of the transmission support are characteristics directly related to the communication medium. In terms of availability, the air is universally available. So, any entity with a transceiver operating in the same bandwidth can both listen and transmit openly, facilitating eavesdrop and anonymous transmission.

VANETs networks scale is increasingly big, being one of the largest ad hoc networks in the world [36]. Thus, creating standardized security policies around the world is a challenging process. As such, finding an entity responsible for managing identification and key exchange worldwide is challenging. These characteristics complicate the usage of traditional cryptography tools. For example, the constant disconnections prevent secure channel creation or the exchange of certificates and CRL.

On the positive side, unlike most mobile networks, VANETs do not suffer from problems such as lack of energy or computing power or storage failure, which allows the implementation of some cryptographic primitives that are more CPU demanding. On the other hand, most of the information must be handled in real-time, which is a big challenge regardless of computing power.

### 2.2.2 VANET Architectures

VANETs are a very heterogeneous type of network and, in the future, vehicles may use multiple types of wireless communications [17]. Thus, considering the diversity in this environment, implementing applications and technologies at the application level should be independent of the underlying medium-access communication framework, allowing their rapid and incremental growth. If possible, the same concept should also be applied to the transport and network levels. This objective can be targeted by applying a concept known as an agnostic middleware communication layer, which can provide independent information management for multiple data sources, allowing the communication over heterogeneous interfaces to be transparently supported over different communication stacks.

Some middleware agnostic are already included in existing standard architectures. These architecture are the following: Wireless Access in Vehicular Environment (WAVE), Communications Access for Land Mobiles (CALM), and European Telecommunications Standards Institute (ETSI) ITS-G5. The mentioned architectures are presented next.

#### WAVE

The WAVE IEEE 1609 family defines an architecture and a complementary set of standardized protocols, services, and interfaces that allows WAVE to operate in a VANET environment [36]. It allows the establishment of communications between V2V and V2I. This architecture, shown in Figure 2.5, also defines the security of exchanged messages. WAVE standards form together the basis for implementing of a comprehensive set of applications in the transportation domain [36]. The WAVE IEEE 1609 standards family is composed as follows [36]: IEEE P1609.0, IEEE P1609.1, IEEE Std 1609.2, IEEE Std 1609.3 IEEE Std 1609.4, Draft IEE P1609.5, Draft IEEE P1609.6 IEEE Std 1609.11, IEEE Std 1609.12.

The problems indicated in Section 2.4 are addressed by WAVE stack, more precisely IEEE Std 1609.2. It defines message formats and processing methods that must be used in order to secure messages. Also, it defines functions necessary to support message security, anonymity, and the privacy of the entities.

This standard [37] defines the message signing as being done using asymmetric cryptography. All the implementation must support Elliptic Curve Cryptography (ECC) and Elliptic Curve Digital Signature (ECDSA) signatures over the two National Institute of Standards Technology (NIST) [38] curves p244 and p256. The



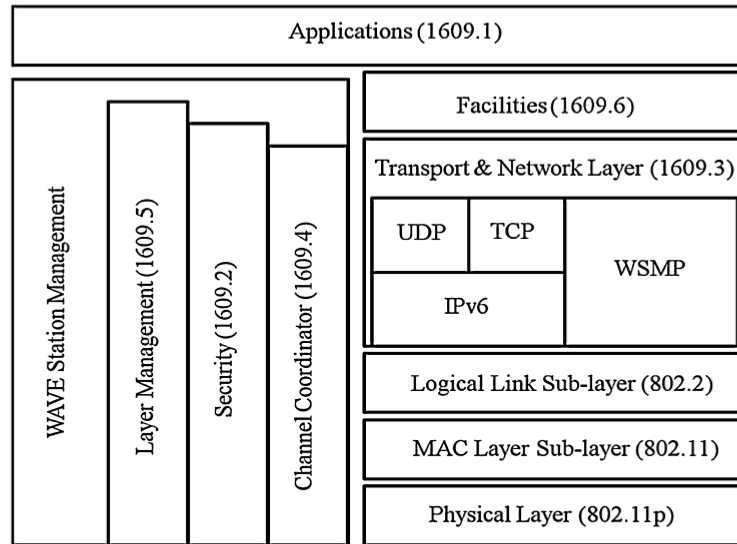


Figure 2.5: WAVE Architecture (From [7])

signing algorithms must be chosen from the following set: `ecdsa_nistp225` or `ecdsa_nistp256_with_sha256`.

To encrypt a message, the sender uses a freshly generated symmetric key. This key is encrypted with the public key of the receiver and added to the message. The only asymmetric encryption algorithm supported is the Elliptic Curve Integrated Encryption Scheme (ECIES) over the NIST curve p256. The symmetric key algorithm supported is Advanced Encryption Standard (AES) in Cipher Block Message Authentication Code (CBC-MAC) with Counter with CBC-MAC (CCM) mode.

If an entity needs to communicate with a remote server with other interoperability requirements, it should use existing protocols as S/MIME's Cryptographic Message Syntax (CMS). If the applications support User Datagram Protocol (UDP), they can also use some application layer security protocol as Datagram Transport Layer Security (DTLS) or an internet layer protocol such as Internet Protocol Security Protocol (IPSec).

The IEEE Std 1609.2 standard also defines the format of certificates and CRL to be exchanged in the network, specifying how each of them must be constructed and signed. Each CA certificate has a scope indicated for which application types it can issue certificates for. It also defines how an entity must proceed in order to obtain a new certificate from the CA.

## Communications Access for Land Mobiles

CALM is a communications architecture defined by the International Organization for Standardization (ISO) Technical Committee 204 - Working Group 16 (TC204 WG16). Its goal is to allow interoperability between Cooperative Intelligent Transportation Systems (C-ITS) stations by abstracting applications and services from the communication layers [39]. The CALM architecture, shown in Figure 2.6, defines a set of standard specifi-

cations [17], including ITS Station Management, Communications Security, Facilities, Station Networking and Transport layer protocols, Communication Interfaces and Services, Interfacing Technologies into ITS Station, Distributed Implementations of ITS Stations, Interfacing ITS Stations to existing communication networks.

Although CALM should theoretically enable vehicles to use any communication media with seamless media handover, supporting the integration of different technologies for Vehicle to Anything (V2X) communications does not present detailed solutions for their simultaneous use by multiple applications in the same ITS station [17]. It is, however, implied to be overcome by implementing network switching inside the OBU.

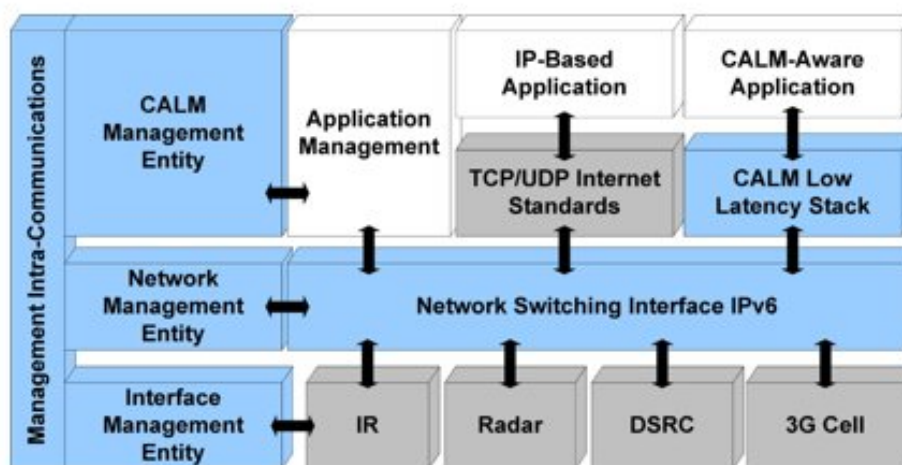


Figure 2.6: CALM Architecture (From [39])

## ETSI-ITS-G5

ETSI ITS-G5 [40] (Figure 2.7) is defined by the standard ISO 21217:2014, which describes the ITS station reference architecture. It consists of six parts: Applications, Management, Facilities, Networking and Transport, Access and Security. However, the standard does not specify whether a particular element should be implemented. It depends on the specific communications requirements.

The Facilities Layer is particularly relevant, providing mechanisms for the agnosticism of the architecture. However, the implementation of such facilities would be very complex, complicating the process of software development and demanding the application development to take into account many technological details of lower layers.

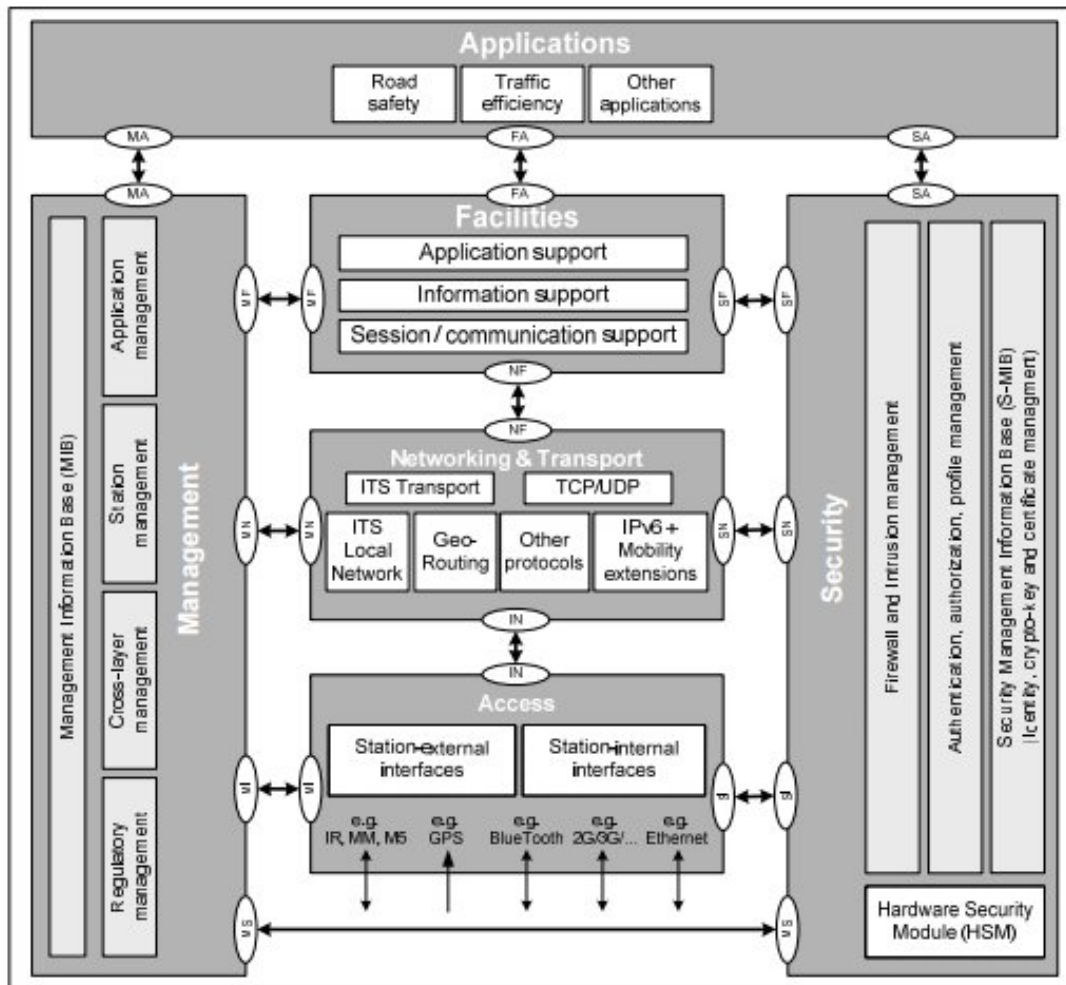


Figure 2.7: ETSI ITS-G5 Architecture (From [40])

### 2.2.3 Medium Access Standards

Standards facilitate the intercommunication and interoperability between components. The availability of standards reduces the cost and time to market. Standardization enables new technologies to be more easily and rapidly implemented. The VANET context affects virtually all the different layers of the Open Systems Interconnection (OSI) model [7]. Currently, the main standards for VANETs communications at the physical layer are Dedicated Short-Range Communications (DSRC) and IEEE 802.11p. These define everything from the medium access and message format to the certificates and their distribution. In the following Sections, these standards are briefly described.

## Dedicated Short Range Communications

**DSRC** is a lightweight communication stack that consists of three layers [41] (see Figure 2.8): physical layers, data link layer, and application layer.

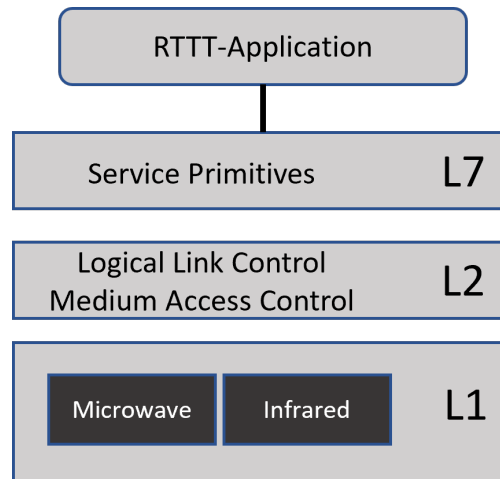


Figure 2.8: **DSRC** Protocol Stack (Adapted from [41])

The physical layer was designed to support different media types, allowing it to be 5.8 GHz microwave transmission, 850 nm infrared, or others [41].

Data Link Layer is divided into two sublayers, the Logical Link Control (**LLC**) and the medium access control layer. The **LLC** provides service primitives for a (un)acknowledged, connectionless data transfer, while the medium access control coordinates the access to the physical medium [41].

Finally, the application layer consists of three kernels elements: the initialization, the transmission of broadcast messages, and the transfer of protocol data units [41].

In the U.S., the band attributed to the **DSRC** is between 5860 and 5892 GHz. It is divided into seven channels (Figure 2.9) of 10 MHz, respectively numbered 172, 174, 176, 178, 180, 182, and 184. The U.S. standard uses channel 178 as the Control Channel (**CCH**), and the others are Service Channel (**SCH**). Channels 172 and 184 are reserved for High Availability and Low Latency (**HALL**) [7].

In Europe, the band is regulated by the **ETSI**, and only channels 180 of **CCH** and 172, 174, 176, 178 of the **SCH** are used [7], as shown in Figure 2.10.

Critical Safety of Life	SCH	SCH	Control Channel (CCH)	SCH	SCH	Hi-Power Public Safety
ch 172 5.860GHz	ch 174 5.870GHz	ch 176 5.880GHz	ch 178 5.890GHz	ch 180 5.900GHz	ch 182 5.910GHz	ch 184 5.920GHz

Figure 2.9: **DSRC** Channels in USA (From [7])

<b>SCH</b>	<b>SCH</b>	<b>SCH</b>	<b>SCH</b>	<b>CCH</b>
ch 172 5.860GHz	ch 174 5.870GHz	ch 176 5.880GHz	ch 178 5.890GHz	ch 180 5.900GHz

Figure 2.10: DSRC Channels in Europe (From [7])

## IEEE 802.11p

IEEE 802.11p is an extension of the IEEE 802.11 protocols created to support vehicular communication in accordance with the DSRC band. It is the hearth of WAVE specification that was adopted by the Society of Automotive Engineers (SAE) for DSRC family of standards and by the ETSI ITS-G5. The medium access and physical definitions are specified in the standard IEEE 802.11p-2010. The IEEE 802.11p is based on the Orthogonal Frequency Division Multiplexing (OFDM) with flow rates of 3, 4, 5, 6, 9, 12, 18, 24, and 27 Mbps and a channel of 10 MHz [7].

## 2.3 VANET Security Requirements

VANETs characteristics make them inherently very challenging. The frequent disconnections in association with the limitations offered by the communication medium complicate communication between vehicles and with the infrastructure. It makes them particularly demanding in terms of security. Secure communications through a HyperText Transfer Protocol Secure (HTTPS) like mechanism is very challenging. In WAVE, for example, instead of establishing a secure channel, the data is encrypted with a randomly generated symmetric key that, in turn, is encrypted with the receiver public key.

According to ETSI TC ITS [42], security requirements that need to be considered in vehicular cooperative systems are the following: authentication, integrity, non-repudiation, privacy, confidentiality, authorization, real-time constraints, and availability.

**Authentication** Involves origin and entity authentication. Every entity within a VANET should be able to verify the authenticity of a message or entity. Messages without a signature should be automatically discarded, preventing attackers from impersonating authentic entities and, for example, influence traffic in its favor.

**Integrity** It should be possible for any entity in the network to verify if a received message is not tampered with. It assures that the final receiver will notice if a message suffered in alterations during its path.

**Non-repudiation** Serves mainly to discouraging entities from misbehaving. If it is assured, no entity should be able to deny performing an action.

**Privacy** Every entity in a VANET should keep its identity private or, at least, only accessible by authorized entities, which is particularly difficult, mainly using traditional authentication methods. The usage of the traditional certificates, for example, allows them to be tracked in the network.

**Confidentiality** Protects data from unauthorized access. When sending a message, only the target destination entity/entities should understand it. It is commonly provided by data ciphering.

**Authorization** It should be possible to define which entities can access a resource or a service. In a platooning application, for example, only some vehicles can join a specific Platooning.

**Availability** Safety applications, for example, need to be constantly available. They need to be able to inform and be informed about possible road problem warnings.

## **2.4 VANET Attacks**

VANETs, as previously mentioned, have some very specific characteristics, which make them especially appetizing for attackers. The attacks to a VANET can be performed by a plethora of individuals or organizations that can differ depending on their motivations. Authors in [5] classify attackers into six categories: individuals, loosely coordinated groups, insiders, adversary organizations, foreign governments, and government agencies. The mentioned attackers can be motivated by different interests, which can be classified into [5] monetary gain, revenge, ideology, belief, intellectual challenge and cyber warfare.

### **2.4.1 Types of Attacks**

VANET environment is intrinsically vulnerable, presenting a large diversity in known attacks. So, Mejri et al. [7] proposed classifying them into categories to better clarify and simplify the attacks. The proposed classification (Table 2.1) is divided into attacks on availability, authenticity and identification, confidentiality, integrity and data trust, and non-repudiation/accountability.

Attack	Classification
Availability	Black hole attack
	Broadcast Tampering
	Denial of service
	Greedy Behavior
	Jamming
	Malware
	Spamming
Authenticity and Identification	...
	GPS Spoofing
	Illusion Attacks
	Key/Cert Replication
	Masquerading
	Message Tampering
	Message Suppression
	Message Fabrication
	Position Faking
	Replay Attack
	Sybil Attack
	Tunneling
Confidentiality	...
	Eavesdropping
	Information gathering
Integrity and data trust	Traffic analysis
	...
	Masquerade
	Message Suppression
	Message Fabrication
Non-repudiation/Accountability	Message Alteration
	Replay
	...
Non-repudiation/Accountability	Loss of event traceability
	...

Table 2.1: VANET Attacks Classification (From [7])

**Attacks on availability** aim to disrupt an entity's ability to receive information. Attackers may try to accomplish so in multiple ways, usually by targeting the communication medium. The most common ways to do so include DoS, jamming, greedy behavior, broadcast tampering, spamming and black-hole.

**Attacks on authenticity** consist of entities trying to identify as someone else to gain control of some part of the network or simply disrupt the normal functioning. Some of the most known attacks include Sybil attack, replay attack, Global Positioning System (GPS) spoofing, and illusion attack.

**Attacks on confidentiality** are performed by ill-intended entities to access information that they should not have access to. The following can be identified as examples: eavesdropping, information gathering, and traffic analysis.

**Integrity and data trust attacks** try to alter or create messages so that the receiver will not differentiate them from authentic ones. Replay attack, masquerade, message suppression/fabrication/alteration are some of the most known attacks.

**Attacks on non-repudiation/accountability** try to erase pieces of evidence that an action was taken. The attacker may, for example, delete evidence of an attack. The most common example is loss of event traceability.

Although not having so much impact on the user or the networks, some other attacks should, nonetheless, be considered. Some of the non-mentioned most common attacks include timing attacks, brute force, and man in the middle.

## **2.4.2 Cryptographic Solutions**

Despite the plethora of existing attacks, security for VANETs is a large area of research, providing techniques and frameworks to secure the communications. Thus, most of the attacks mentioned in Section 2.4.1 can be prevented by implementing cryptography or changes in the transmitting devices. Table 2.2 links some of the attacks for VANETs with the technique to preventing them. These are not an overall solution, but only for each individual problem it tackles.



Attack	Solution
Jamming	Switch the transmission channel and use frequency hopping technique FHSS [43]. This requires a modification to the current standard which only allows OFDM
Denial of Service (DoS)	Use bit commitment and signature based authentication mechanisms.
Message tampering/ suppression/- fabrication/alteration	Use vehicular PKI or zero-knowledge techniques [44] for authentication and signing. Establish group communications causing that intruder cannot communicate with the group.
Timing attack	Use time-stamp techniques. It may be hard to accomplish due to synchronism, mainly in a VANET context.
Eavesdropping	Encrypt data
Traffic analysis	Encrypt data. Use algorithms such as VIPER [45] for V2I
Brute Force	Use strong encryption keys and key generation algorithms which, are unbreakable in a reasonable time.
Key and / or Certificate replication	Use certified and disposable keys. Check the validity of digital certificates in real time via CRL. Use cross certification between the different CAs involved.
Sybil attack	Deploy a central Validation Authority (VA), which validates entities in real time.
Illusion attack	Protect hardware and software to be only accessible by authorized people. Every reading or update operation received from sensors must be authenticated and verified. Values and protocols should be stored in piratically protected hardware.
GPS spoofing or position faking	Bit commitment and signatures should be used, positioning systems should only accept authentic data.
Replay	Packets that may be replayed should be timestamped. There may be the problem of synchronization between entities.
Man in the middle	Use strong authentication methods, for example, digital certificates or zero-knowledge
Node impersonation	V2V and V2I communications should be made using variable MAC addresses and IP. Authentication through digital certificates. Use distance bounding protocols based on bit commitment and zero-knowledge [46]
Greedy, blackhole, grayhole, sink-hole, wormhole, malware, masquerading, spamming, tunneling	There isn't a real cryptographic solution for these attacks but, the usage of digital signatures can reduce the attack effects. Using piratically protected hardware can help prevent them.
Broadcast tampering	This can be made by a legitimate node. So, cryptographic tools may not be of any help. Non-repudiation mechanisms may discourage it.
Loss of event traceability	Same as the solution proposed for the illusion attack
Tracking, Social engineering	Use variable MAC and IP addresses. Their allocation must be done by a robust algorithm

Table 2.2: Solution for VANET Attacks (From [7])

Most of the presented solutions make use of mechanisms such as digital certificates, ciphers, times-

tamping, and variable MAC and Internet Protocol (IP) addresses. Also, modifications to the transmission channel are proposed for the jamming attack. The Sybil attack is challenging to solve as legitimate entities can perform it. The proposed solution is the deployment of a VA. However, these need a connection to the infrastructure. The VA may become the target of attacks due to its privileged role.

## 2.5 Security Models for VANETS

VANETs are a very challenging and diverse type of network. The heterogony of entities and their constraining characteristics complicates the implementation of an overall security model. Nonetheless, the research on this subject is extensive, providing multiple solutions. ETSI [40], for example, has proposed a high-level security framework [1]. However, this approach is too focused on message-level security for Context Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM). So, it does not consider the integration of secure transport and network services, like IPSec services [47].

Furthermore, the security models proposed by Institute of Electrical and Electronics Engineers (IEEE) WAVE and ETSI TC ITS have not considered session-based security associations, and messages are protected individually using a PKI. However, this provides a security scheme only valid for broadcast scenarios and cases with low V2V and V2I traffic volumes, which does not cover all vehicular applications and services.

However, due to the VANET characteristics and its plethora of attacks and attackers, the consensus for the ideal security model is far from being achieved. Also, the existing solutions have their own advantages and shortcomings, each better suited for different situations, difficulting the decision. Bariah et al. [5] summarize the most common approaches: PKI-based, ID-based cryptography and situational modeling-based. Table 2.3 summarizes the main challenges in each of the solutions.

<b>VANET Security Mechanism</b>	<b>Challenges</b>
PKI	Comm resources by CRL, location privacy
ID-Based	ID privacy
Situation Modeling	Complexity of various situations

Table 2.3: VANET Security Mechanisms Challenges (From [5])

### 2.5.1 Based on Public Key Infrastructure

The more traditional PKI-based models [2] [3] [48] [49] [50] [4] already provide authentication, key exchanging mechanisms, and certificate management tools (signing, issuing, check, maintenance, audit,

renewal, cancellation, etc). PKIs, bind certificates to public keys and the identification of the owner providing, a way to sign and distribute keys.

PKI may be very useful as its certificates have all the needed parameters for secure communications. Any entity can verify if they were issued by the correct CA and check if they are valid. They also contain the public key of the sender, which allows for signature verification. Moreover, the certificate public key can be used to cipher the response to be sent back. However, the key also poses a challenge. Without knowing it, the establishment of secure communication cannot be done. Needing for the key of every receiver to be known beforehand can be extremely complicated in a VANET scenario.

VANETs communications may be sparse, which can create obstacles for a PKI implementation. Mainly due to key and CRL distribution. Some solutions aim to solve that problem by implementing a dynamic key distribution [3] or region-based certificate distribution[48].

Also, PKI leaks information about the identities of the users. Attackers can use the certificates to track users in the network. Some research works propose solutions to tackle the privacy limitations in PKI. One of the most common solutions is the distribution of Pseudonym Certificates (PCs) to provide anonymity and privacy protection [51, 52]. Which, in turn, also has its limitations. The introduced Pseudonym CA (PCA) will be able to track entities identification. Moreover, its encryption mechanism does not have the needed capabilities to enable broadcast or multicast scenarios easily. PKIs use the public key of the receiver entity to encrypt the data which means that only one entity will be able to decipher the data. Multicast scenarios will need group key exchange to be done previously.

## 2.5.2 Identity-Based Encryption

ID-based cryptography [53] enables entities to cipher data and verify signatures, enabling the creation of a secure communication channel without needing previous key exchange. Moreover, any entity can decipher the information, regardless of the source, provided it is its destination.

This scheme is similar to public key cryptosystems [53] but, instead of generating a public/private key pair, the public key is based on identity. This can be anything from a username to a social security number or an email. The private key is generated by a central TA. The usage of a TA is needed due to the properties of the public key. There is nothing secret about an entity id so, if everyone had the power to create secret keys for itself, it would have the power to create any key [53]. Also, the TA generates a set of public parameters used to generate the real public key from the ID. These parameters can be distributed freely to everyone.

The TA is a key element of this scheme, making it a potential target for attacks. Compromising it may compromise every entity in the system.

ID-based cryptography has the advantage of using entity identification (email, network address, etc.)

to verify and sign messages [5]. The ID-based mechanism can be a good replacement for a VANET context since it does not require certificate fetching and verification. It outperforms PKI in some aspects, such as communication bandwidth and storage, as it does not need CRL and certificates distribution. Bariah et al. describe the main drawbacks of this method as being: guaranteeing user privacy [5] and having a single point of failure, the TA.

### 2.5.3 Based on Situation-modeling

Mechanisms based on situation modeling analyze routines of driving trends and try to generate the most adequate security model for that specific situation and, thus, increase security by managing its nodes [5]. It assumes that vehicle driving tends to be fixed on some routes, as, for example, commuting to work and shopping habits. The major concern with these solutions is the uncertainty of everyday vehicle movement, causing it to create many models.

Huang et al. [6] propose Situation-Aware Trust (SAT) model to address trust issues in VANETs. It is a combination of attribute-based policies, a proactive trust model, and an email-based social network. The attribute-based policies are used in an ABE scheme, using its properties to provide data access control. Data can be ciphered to only be accessible by elements of a certain company.

SAT provides proactive trust by establishing it before entities meet. To do so, it distributes certificates to vehicles before the any communication takes place. It also provides a framework that helps reduce unnecessary trust establishment actions between entities, which requires that each vehicle predicts the potential vehicles it will encounter by distributing its trajectory [6].

Additionally, it adopts a social network trust to provide a rapid establishment of trust, particularly an email-based social network trust. Which already have some properties that make them advantageous to other social networks; for example, they already provide an intrinsic layer of security. E-mail IDs are unique, and they can potentially provide a very large trust database, etc.

## 2.6 Intrusion Detection Systems

Attacks, such as those performed by authenticated entities, cannot be prevented using traditional security mechanisms [54]. IDSs while not being able to stop attacks, can provide an extra layer of security by detect unpreventable attacks and trigger a response minimizing the nefarious effects on the targeted system [9]. Intrusion detection is made in three phases [9]: data collection (logging system calls, recording traffic received, etc.) [54], analysis phase (pattern matching, statistical analysis, data mining, etc.) [54], and the response. Depending on the detection technique used, IDSs [54] can be classified into signature-based,

anomaly detection systems, specification-based systems, and reputation management.

In signature-based intrusion detection, the **IDS** tries to match the collected data with known attack patterns. It has a low False Positive Rate (**FPR**) but only detects already known attacks. This type of detection also needs a large storage capacity due to the large amount of data of known attacks needed. It is most effective when used in Wireless Local Area Networks (**WLANs**) [54].

Anomaly detection systems work from the collected data history (unlabeled) or a set of training data (labeled). These two types are called unsupervised and semi-supervised, respectively. An anomaly is any behavior different from the usual and preestablished. These detection types are most effective when used for mobile telephony base stations. They have well-established and predictable behaviors, which are favorable to anomaly detections systems [54].

Reputation management is most useful to detect selfish behavior. A reputation manager is needed for this detection system, as nodes can collude to improve their rating. They are most effective in ad hoc networks. The design of ad hoc networks difficult the implementation of anomaly-based systems and, due to its maintenance obstacles, reduces signature-based systems effectiveness [54].

Specification-based intrusion detection systems detect anomalies at the system level, unlike the anomaly detection systems that analyze user profiles and data flows. It has a low false-negative rate, which is a major advantage, and the system is immediately effective because they do not need a previous training phase. However, they only detect known bad behavior, and massive efforts are needed to generate a formal specification. They are most effective for Wireless Sensor Networks (**WSNs**) and unattended Cyber-Physical Systems (**CPSs**) due to their predictability and limited resources [54].

The **IDS** performance is usually measured using three metrics [54]: **FPR**, False Negative Rate (**FNR**), and Detection Rate (**DR**). The **FPR** measures the number of times the detection system identified a normal behavior as being an intrusion. The **FNR** measures the number of anomalies that were not identified. Finally, the **DR** (also known as true-positive rate) measures the correctly identified threats.

According to Erritali et al. [9], **IDS** architectures can be classified into three different types: standalone, cooperative and distributed, and hierarchical. Standalone architectures do not need to exchange data, as each node only relies on itself. The cooperative and distributed architectures nodes exchange information between themselves cooperating to detect intrusions. The messages exchanged can increase network traffic, degrading network performance. The hierarchical architecture divides the network into clusters. Each cluster has a cluster Head (determined by a clustering algorithm), reducing the number of communications needed.

Traditional **IDSs** assume that the behavior of an intruder will be noticeable from the normal network functioning. Signature detection systems compare the behavior with known attacks, assuming that there will not be new attacks. Anomaly detection systems focus on detecting significant deviations from normal behavior. However, the definition of the criteria to decide what constitutes misbehavior is difficult to attain [10]. The usage

of intelligent algorithms enables the IDSs to learn from previous attacks and detect new attacks. Recently, IDSs have been used in VANETs to mitigate some attacks that cannot be prevented by using cryptographic tools.

## 2.7 Machine Learning

ML, use data mining techniques to infer knowledge from collected data. These techniques are used to find patterns in already gathered data, even if it is not particularly new. For example, data collection of user preferences can be mined to find behavioral patterns and find the likelihood of a client buying a product or a service [55].

The ML technique can be classified into[56] supervised, unsupervised and semi-supervised based on the learning methods. The supervised and unsupervised classification methods are described in Section 2.7.1 and Section 2.7.2, respectively. The semi-supervised methods are a hybrid of the two, where the data can be of either type.

### 2.7.1 Supervised Learning

Supervised learning assumes that each instance has a correspondent label. This classification model needs a set of labeled training data, which can be used as an example. Gathering labeled data can be time-consuming and hard to obtain [57]. The following are examples of supervised learning algorithms: Decision trees, Artificial Neural Networks (ANN), Support Vector Machine (SVM), fuzzy logic, genetic algorithms.

**Decision Trees** use a combination of nodes and branches. Each node represents an instance to be classified, and the nodes represent the values they can have. Decision trees classify the instances by starting at the root node and sort them by their values [58]. There has been extensive research in this area, mainly related to the construction of optimal binary decision trees.

The trees construction is started by finding the feature that best divides the data, the root node. The same process is repeated until one of the following conditions is met [59]: all the set instances belong to a single class, the maximum tree depth is reached, or the best splitting criteria is less than a pre-defined criterion.

**Artificial Neural Networks** are computational networks created to simulate human beings or animal biological neural behavior [60]. They are designed to learn in a similar way to the human brain. These acquire knowledge through a learning process, storing it in the connections between several cells.

The main advantages of ANNs are their capacity to model complex relationships that are implicitly in the data, robustness against wrong and missing data, and the possibility to obtain results, as long there is data but without the need to necessarily have any previous assumptions about the problem [60].

There are several types of ANNs, including the Multi-Layer Perceptrons, successful since the 80s, after introducing the famous Backpropagation algorithm. These depend on three fundamental aspects [58]: input and activation function, network architecture, and each the weight of each input connection. The training of the networks is made by comparing the output of the ANN with the desired value and adjust the weights.

Recently, since the 2010s, there was a growing interest in more complex ANNs with several intermediate layers designated for Deep Learning [60]. In fact, Deep Learning is currently considered the best automatic learning model. It has obtained the best results in computer vision or speech recognition competitions. However, Deep Learning demands high volumes of data, Big Data, and computational power.

**Support Vector Machines** tries to find a hyperplane that divides the data into two classes. The division tries to maximize the margin on either side of the hyperplane, creating the largest possible distance between the hyperplane and instances on either side [60].

The model training involves using large matrix operations and time-consuming numerical computations as it is done by solving the Nth dimensional Quadratic Programming (QP) problem, being N the number of samples. A Sequential Minimal Optimization (SMO) algorithm can be used in order to improve performance. It solves the QP problem without adding any matrix storage by decomposing the problem into QP smaller sub-problems [60].

The SVM algorithm ends when a global minimum is found, avoiding local minimums (Unlike other algorithms such as neural networks) [60].

## 2.7.2 Unsupervised Learning

Unsupervised machine learning has datasets containing only unlabeled data. These classification methods can find patterns in unstructured data. For example, clustering and dimensionality reduction can be mentioned [61]. K-means, self-organizing maps, k-medoids, and Bayesian clustering are examples of unsupervised learning techniques[59].

**K-Means** divides a set of n d-dimensional points into K clusters [62]. It calculates the squared between the mean of a cluster and its points and tries to minimize it. It can only converge to a local minimum due to its characteristics. The minimizing function is an NP-hard problem, which makes it a greedy algorithm. As it is described by Jain et al.[62], the main steps of a K-means algorithm are:

- Select an initial partition with K clusters. While cluster membership is no stabilized, repeat steps 2 and 3;
- Assign patterns to its closest cluster center generating new partitions;
- Compute new cluster centers.

**Self-organizing maps** are a type of ANN where neighbor cells compete by means of lateral interaction, which allows them to develop into specific detectors of different patterns [63]. These ANN cells become tuned to various input signal patterns through an unsupervised learning process.

If self-organizing maps are used as a pattern recognition ANN, they can be fine-tuned using supervised learning. When used in classification problems, their decision accuracy can be increased using LVQ fine tuning [63].

**K-medoids** clustering chooses k members from the data set. These members are called medoids and are defined as the cluster object in which the average dissimilarity to all cluster objects is minimized. The clusters should structure the data in such a way that similar objects are in the same cluster, and different objects should be in different clusters [64].

## 2.8 Platooning

Platooning is a concept that consists of several vehicles traveling very close to each other in groups with constant speed and gaps between them. It enables the enhancement of safety, traffic flow, and highway capacity while also providing drivers with a more convenient and comfortable driving experience. Furthermore, it helps to save energy and fuel, reducing emissions. An example of platooning is presented in Figure 2.11 by the trucks between the red brackets.

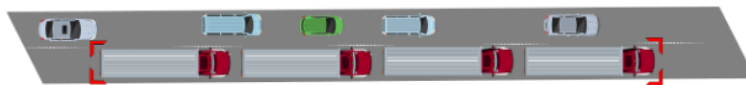


Figure 2.11: Platooning of Vehicles (Trucks Between Red Brackets)

A Platoon of vehicles may be composed of several types of vehicles, from trucks and buses to passenger vehicles [65]. A vehicle belonging to a Platoon can perform one of two roles - Leader or Follower. The Leader is the one that controls all the maneuvers of a Platoon. It can adjust multiple parameters of the Platoon, including speed and headway distance. The remaining vehicles are the Followers, these, as the name indicates, follow the Leader, and thus, their drivers will be able to undertake other tasks such as using a mobile phone.



The simplest way of implementing a platooning is through V2V communications, where the vehicles only share information with their predecessor. More complex and advanced implementations allow vehicles to share information with members not in the line of sight, providing the driver with situation awareness feedback. The shared group information may allow the Followers to better predict Platoon behavior, helping stabilize the Platoon.

For security reasons, the size of the Platoon is limited. It can be defined as the distance between the first and last vehicle of the Platoon, but it is usually the maximum number of vehicles. The maximum size of the Platoon was first defined as 20 [66], but some more recent studies advise it to be no bigger than 15 vehicles.

From a communication standpoint, platooning requires an efficient Platoon Management Protocol (PMP) [12] that specifies all the required maneuvers and proper communication behaviors.

### 2.8.1 Platooning Maneuvers

As described in [12], there are five possible maneuvers are: Create, Join, Leave, Dissolve and Merge.

The first, **Create**, is a process with which a vehicle can create a new Platoon. Only the Leader can perform this maneuver, and the process is the following: The Leader creates the Platoon and starts broadcasting its existence.

The **Join** maneuver is initiated by a Follower wanting to join an existing Platoon. This maneuver can happen in two different ways, in the rear or any place in the middle of the Platoon (more complex) chosen by the Leader. This process is started by sending a Join Request to the Leader. If the maneuver is possible, the Leader responds with a Join Acknowledgment. Otherwise, the Leader sends a Join Reject indicating the reason for the rejection. The Joiner changes to the correct lane and position and informs the Leader with a Distance Achieved if all goes well. The Leader then opens a space big enough for the new vehicle to join, sending an Adjust Gap message. When the Followers achieve the desired gap adjustment, they send an Adjust Gap Acknowledgment. The Leader can then send a Start Maneuver message informing the Follower that the maneuver can be performed. The Follower enters the Platoon and informs the Leader sending a Maneuver Completed message. Finally, the Leader sends an Update message to the Follower with the new Platoon information.

The **Leave** maneuver is invoked by any Follower that desires to exit the Platoon. This maneuver can only be performed by one vehicle at a time and only if it is authorized by the Leader (Except exceptional cases, vehicle failure, emergency exit, etc.). To do so, the Leader starts by informing the Leader by sending a Leave Request. The Leader adjusts the gaps between the Followers in order for the vehicle leaving to be able to do so safely by sending an Adjust Gap message. The Followers respond with an Adjust Gap

Acknowledgment message. When all the vehicles have successfully adjusted their gaps, the Leader can send a Start Maneuver for the leaving vehicle. After leaving the Platoon, the leaving vehicle informs the Leader with a Maneuver Completed message. The maneuver is terminated with a Platoon Update message to all the Followers, updating the Platoon parameters.

The **Dissolve Maneuver** is performed by the Leader when it decides to disassemble the Platoon. The Leader informs all the Followers about its decision with a Dissolve Request and only dissolves the Platoon when it receives the Dissolve Acknowledge from all the Followers.

The **Merge Maneuver** is performed between two Platoons that want to join. The only actors in this maneuver are the Leaders of the two joining Platoons. To better explain the process, the front Platoon will be called A and the rear B. Both Leaders send Merge Request messages every 10 seconds. If Leader A receives this message, it will respond with a Merge Acknowledgment. The Leaders exchange Platoon Info messages with information about their respective Platoons. Leader A sends an Adjust Gap message to Leader B. Leader B moves closer to the rear of Platoon A. The Leader from Platoon B sends an Adjust Gap Acknowledgment to Leader A and a New Leader to its Followers. Leader B becomes a Follower.

## 2.8.2 Platooning Security Requirements

Platooning applications have very specific security requirements due to their underlining environment, VANETs, and the complex nature of the application itself. The best way to characterize the security requirements is to divide them into three layers: Communication, Application, and System [67].

### Communication Channel Security Requirements

The Platoon members need a channel that implements a basic set of secure properties to communicate securely with each other. The security channel can be implemented by the lower layers or, if not possible, by the application itself. This channel can be similar to [HTTPS](#)/Transport Layer Security (TLS) on the internet. It should provide availability, integrity, authentication, and non-repudiation. Additionally to the indicated mandatory requirements, the channel should also provide privacy and confidentiality. Although the last two are not required as the application can still function without them, the members of a Platoon should be able to keep their privacy intact. Also, the messages exchange should be kept confidential, at least in some cases. The Platoon may be implemented as a private service allowing only entities belonging to that Platoon to keep track of their actions.

Integrity is a crucial security requirement. All entities should be able to verify if the sent message is the same as the one received. Moreover, without integrity, it is not possible to guarantee some of the other requirements as it is the case for authenticity or non-repudiation.

At this level, authenticity only refers to exchanged messages. Meaning, it should be possible for all entities to verify if the message was sent by an authentic entity.

In this type of applications non-repudiation, can be used to track attacks or responsible entities. Any message or action should be impossible to deny and verifiable by any entity.

With regards to privacy, although it is not a requirement, it should be provided. All entities should be able to keep their information private, at least inside a specific Platoon. If it is possible for an attacker to track messages, it may be possible to track entities in the network and discover more valuable information.

Similarly, confidentiality should also be provided. Although it may be needed to maintain privacy, some entities may desire to create a Platoon as a private service and keep the information exchange in a Platoon secret. Also, it may be easier for attackers to disturb the Platoon if they know all the parameters and messages exchanged inside a Platoon.

### System Layer Security Requirements

In [67] are described system layer attacks. These can tamper with the vehicle hardware or software and be carried by a malicious insider. These can be prevented by implementing tamper-proof sensors in the vehicle, tamper evidence, and detection mechanisms. These are out of the scope of this thesis.

### Application Layer Security Requirements

A different set of security requirements are identified at the application layer, authentication, non-repudiation, and membership management.

At this level, authentication refers to the drivers. The driver must prove his identity to the vehicle and to the Platoon itself. Depending on the platooning implementation, only a specific driver in a specific vehicle may join a Platoon.

Asplund et al. [68] describe six different scenarios that may happen if tight membership management is not maintained:

- **Scenario 1:** Vehicle B joins a Platoon created by vehicle A. B then creates a new Platoon joined by C without A's knowledge. It will make A only see the Platoon as being composed of himself, and B and C will see the Platoon as C and B, which are inconsistent.
- **Scenario 2:** Vehicle A and B form a Platoon. Vehicle C creates a new Platoon and claims to be in B's position. If a new vehicle (D) joins, it will be in a two-car Platoon formed by C and D, when in reality, it is in a three-car Platoon made by A, B, and D.

- **Scenario 3:** Vehicle A is a non Platooning vehicle. Vehicle B creates a Platoon and announces to be in A's position. If some vehicles C and D join the Platoon, they will think they are being led by vehicle B when in reality, the front vehicle is vehicle A.
- **Scenario 4:** Vehicle B impersonates two different vehicles and sends requests to A in their names. Vehicle A will think that the Platoon has 4 vehicles when only himself and vehicle B are part of it.
- **Scenario 5:** Similar to the previous scenario but, in this case, the Platoon is joined by a vehicle C that does not have platooning capabilities.
- **Scenario 6:** Similar to scenario 1 but, in this case, nodes B, C, and D are colluding. B pretends to be the tail, C is silent, and D pretends to be the Leader. So, node A thinks it is a two-vehicle Platoon.

Of the indicated scenarios, the most problematic are 1,2 and 6. In these, vehicles will not have the full knowledge of the complete Platoon size. These can be mitigated by keeping tight membership management. Vehicles should be able to verify the identity of their neighbors (or at least to physically identify the Platoon members). Also, they should exchange messages between all the members of a Platoon to verify the consistency and construct a Platoon model. It should be possible for vehicles to detect nodes trying to impersonate multiple nodes to prevent Sybil attacks.

# Chapter 3

## Related Work

The usage of [IDSs](#) in [VANETs](#) is a relatively new field of work, having only a few studies in the literature from 2010. Thus, a survey enabling the identification of the most common approaches, used algorithms, tools, and datasets is needed. One of the most accepted ways to conduct an extensive review of the existing literature is using an [SLR](#)[\[11\]](#).

[SLR](#) is a means of evaluating and interpreting all available research work relevant to a research question or area. This type of research follows a well-defined methodology with a predefined strategy allowing the identification of works relevant to the study being carried out, decreasing the probability of biased results (although not protecting against publication bias in the primary studies) [\[69\]](#).

The purpose of this review is to perform a thorough and well documented collection of the research works existing in the literature. These evaluate the feasibility of implementing an [IDS](#) for [VANETs](#), focusing on the ones taking advantage of intelligent algorithms. Moreover, this study should help the identification of which types of [IDSs](#), [ML](#) algorithms and datasets are most commonly used. Additionally, these works should provide insight into test and evaluation methodologies.

### 3.1 Systematic Literature Review Methodology

An [SLR](#) follows a strict methodology with a predefined search strategy providing an efficient and exact way to gather and evaluate existing works, thus being easily replicated and peer-reviewed. The [SLR](#) presented in this work follows the methodology proposed by [\[69\]](#) and is shown in [Figure 3.1](#).

The methodology is comprised of 6 main steps: define the Research Questions ([RQs](#)), specify the literature sources and search string, select relevant studies, assessing the quality of the studies collected, extract the data from the studies and, finally, synthesize the collected data.

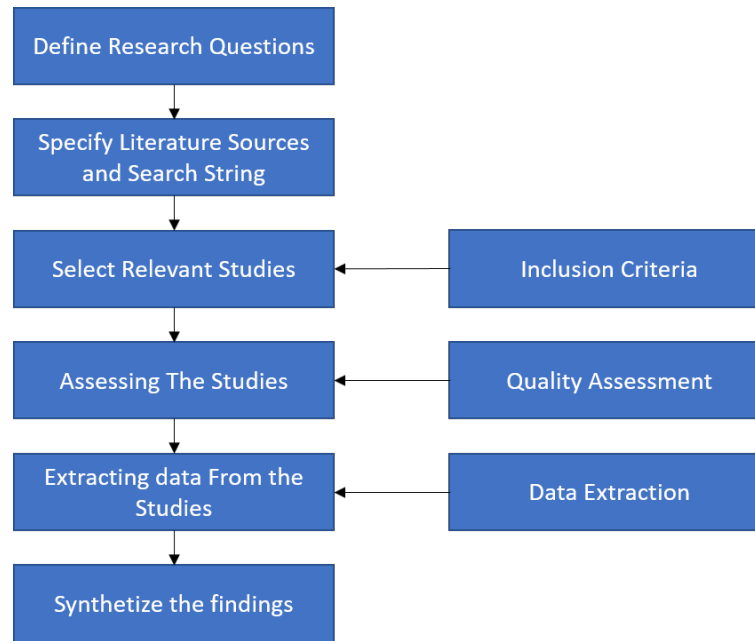


Figure 3.1: SLR Methodology (Adapted from [11])

### 3.1.1 Research Questions

This review aims to identify the state of the art of IDSs for VANETs, mainly the ones that use ML. The RQs defined intend to survey the main characteristics of those IDS including, the IDS design and location, the attacks it targets, the simulators used, the datasets, and the evaluation metrics. The RQs are the following:

- **RQ1:** How can an attacker target VANETs? This question aims to find which types of attackers exist in a VANET environment and how they can target their nodes.
- **RQ2:** Can IDSs detect attacks targeting VANETs? Mainly the ones with no cryptographic solutions. Some IDSs can detect attacks in more traditional networks but, VANETs have a different structure and communication types. This question aims to find if IDSs can be used in these types of networks, with a special interest in attacks not preventable by other tools.
- **RQ3:** Which kind of IDS can be used in VANETs? This question aims to find which type of IDS is best suited for this environment.
- **RQ4:** Can ML algorithms improve IDSs to detect attacks, and which types of ML are the most suited? VANETs have characteristics that can facilitate several vulnerabilities. Thus, there can be a plethora of different types of attacks and attackers. This question aims to find if the usage of ML algorithms can help to detect attacks.
- **RQ5:** How can an IDS be tested and evaluated?

- **RQ6:** Which metrics can be used to evaluate the IDS performance? Question 5 and 6's main goal is to find how an IDS can be tested and evaluated and which metrics can be used.

### 3.1.2 Literature Sources and Search String

Using the previously defined RQs, a search string was created using the following method [69][11]. First, the main search terms are derived from the RQs. Then, using documents already analyzed, more search terms are collected. The next step is to find synonyms and alternative spelling for the major terms already found. Finally, the string is built by joining the terms found. Boolean ORs are used to join alternative spellings and synonyms and ANDs to join major search terms. These can be found in Table 3.1.

Table 3.1: Search Terms, Alternatives and Synonyms Used in the SLR

Search Terms	Synonyms and Alternatives
VANET	VANETs; Vehicular ad-hoc network;
IDS	Intrusion Detection System; Anomaly Detection
Machine Learning	Intelligent Algorithms
Attack	Attackers; Intrusions; Intruders

The search terms found were used to build a Search Query (SQ) and perform a preliminary search using some of the major databases, namely, IEEE, Science Direct, and ACM. The obtained documents were filtered, keeping only those mentioning VANET and IDS. The studies resulting from this research were:

- IEEE - 6 Papers
- Science Direct - 6 Papers
- ACM - 2 Papers

Due to the small volume of results obtained, it was decided to increase the number of databases. And thus, the research for new databases was started.

However, two issues were encountered: the query language is not the same across all platforms; the number of studies resulting from some surveys was too large to be manually analyzed. During this process, an aggregator platform was found, Crossref [70]. This is a non-profit organization that provides a free-to-use platform. It contains studies from the most well-known databases. Crossref provides an easy-to-use REST API that allows to search documents based on their titles and metadata. Moreover, there are some well-documented libraries available implemented in several languages.

The main issue with this platform is the lack of support of Boolean AND, performing an OR operation with all terms. Thus, any study that contains any of the search terms is returned. Therefore, some of the previously defined terms may not make sense in this paradigm. For example, using the acronym for Intelligent Transportation System, ITS, would return all documents containing "its" in their metadata.

To accommodate this new perspective, a new [SQ](#) was built, **"vanets vanet vehicular ad hoc networks network intrusion detection system systems anomaly anomalies intelligent"**. Although simple, this search string includes all the terms that should be present in all the resulting studies.

### 3.1.3 Studies selection

As Crossref only performs Boolean OR, using the [SQ](#) 6,239,042 studies were returned. In this search, no filters were used, and all the supported databases were searched. As the volume of the returned studies was too large to be manually analyzed, an automatic tool was built. A Python application was built using the "habanero" library [71], a well-documented library that allows easy Crossref access.

This automated tool receives an [SQ](#) and uses it to execute multiple requests to the Crossref platform until all the results are returned. A filter was implemented to reduce the volume of returned studies. Its purpose was to provide a boolean AND that Crossref does not have. So, two sets of terms were built. These were used to perform boolean OR inside each group and boolean AND with the terms of the other group. Both groups of terms are shown in Table 3.2.

Table 3.2: Groups of the Search Terms Used in the [SLR](#) Automated Tool

1 <sup>st</sup> Group	2 <sup>nd</sup> Group
vanet	ids
vanets	idss
vehicular ad hoc network Learning	intrusion detection system
vehicular ad hoc networks	intrusion detection systems
vehicular ad-hoc network	intrusion detection
vehicular ad-hoc networks	intrusions detection
intelligent transportation system	anomaly detection
intelligent transportation systems	anomalies detection
	intelligent detection

After filtering the obtained studies using the described terms terms, 41 papers remained. All the returned results were stored in a Mongo database to provide easy access and manipulation of the results.



### 3.1.4 Inclusion/Exclusion Criteria

Using as an example the parameter defined by the authors in [11], the following inclusion criteria were defined:

- If a study has a journal and a conference version available, only the journal version is kept;
- If a study has several versions published, only the most recent is kept;
- If a study exists in more than one source, only one copy is included;

In addition to inclusion criteria, excluding criteria is also defined to exclude ineligible studies. The defined exclusion criteria are the following:

- Only papers available to download are kept. Some papers have restricted access and so will not be included;
- Only studies from conferences or journals that are indexed in Scopus are included. This criterion is meant to only use studies from reliable sources;
- Studies that do not consider VANETs;
- Studies that do not consider MLs.

The inclusion and exclusion criteria were applied to the 41 studies. 12 of the documents filled the criteria and were kept. Two of the documents found were reviews on IDSs, "A Survey on Intrusion Detection Systems and Honey-pot based proactive security mechanisms in VANETs and VANET Cloud" [46] and "A review and classification of various VANET Intrusion Detection Systems" [72]. Of these two papers, only the first was considered. In this study, an in-depth SLR of IDSs for VANETs is made. Therefore, its relevant studies were collected and used to enrich the SLR. Since the second document does not apply any methodology to the review it makes, it was discarded.

After using the same criteria on the survey papers and removing the repeated papers, 22 papers remained in total. Figure 3.2 shows how the papers are distributed by year. It is noticeable an increase in studies from 2013 with a drop in 2017.

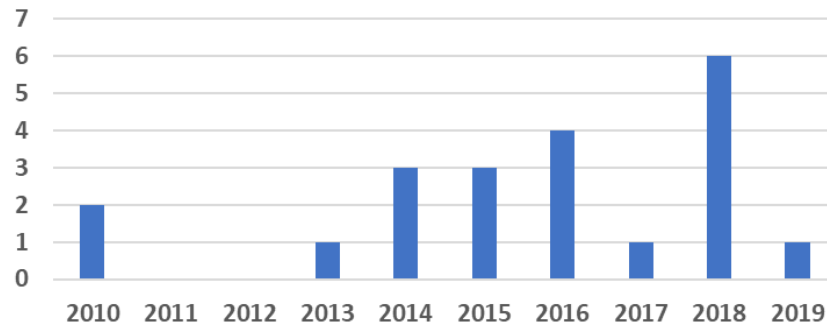


Figure 3.2: Number of Studies on Intelligent IDS Published Per Year

### 3.1.5 Data Extraction

At this stage, the objective was to extract specific information about the gathered studies. A table with parameters to be collected from the studies was distributed to other reviewers to eliminate possible bias and increase the precision in the data collection. These parameters are network simulator, traffic simulator, type of IDS, detection type, ML algorithm, which attack types were targeted, which datasets were used and where the IDSs were located.

In Figure 3.3, the used network simulators are presented. The most commonly used network simulator is Network Simulator 2 (ns-2), followed by Network Simulator 3 (ns-3). These are two of the most popular network simulators. They provide the majority of the network implementations, including WAVE and ITS-G5, which are open-source and highly customizable. Some studies use Matlab or their own simulator, and others just use the data directly from the datasets (usually from datasets publicly accessible). Unfortunately, some of the papers do not specify which simulator was used.

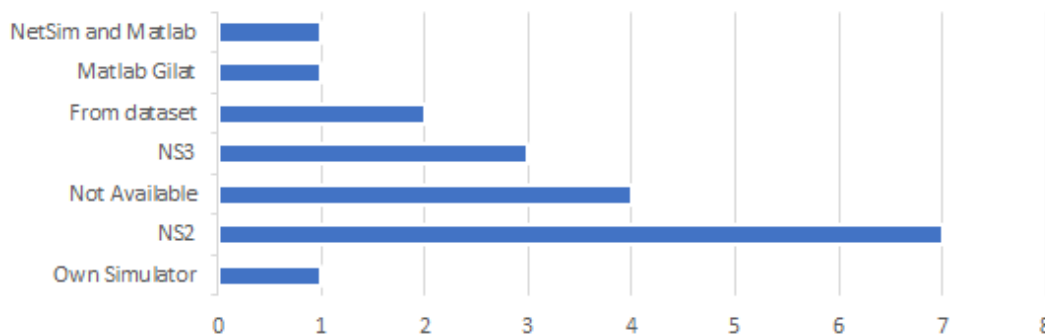


Figure 3.3: Types of Network Simulators Used in Intelligent IDSs Related Works

The most used traffic simulator was SUMO, which in some studies is used in combination with MObility VEHicles (MOVE), as shown in Figure 3.4. From the information collected, only one more traffic simulator was

used, VANET Mobisim. The remaining studies used data from a dataset or did not specify which one was used.

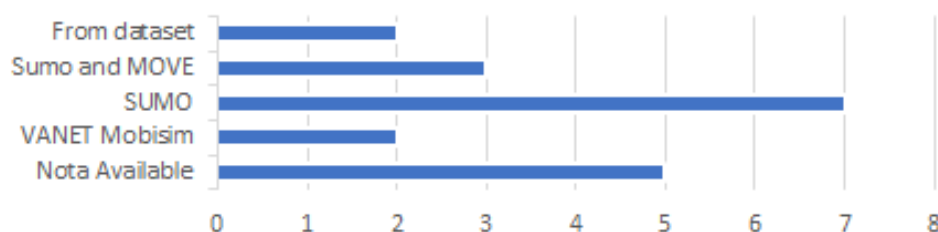


Figure 3.4: Types of Traffic Simulators Used in Intelligent IDSs Related Works

One of the key aspects of this research was to find which datasets were used in each study. These are shown in Figure 3.5. Most of the studies collected their datasets from the simulation. Some were obtained from the trace file generated by the network simulator ([ns-2](#) and [ns-3](#)) and others from values extracted during the simulation. Some studies used the existing datasets, the Kyoto dataset [73] and NSL-KDD [74].

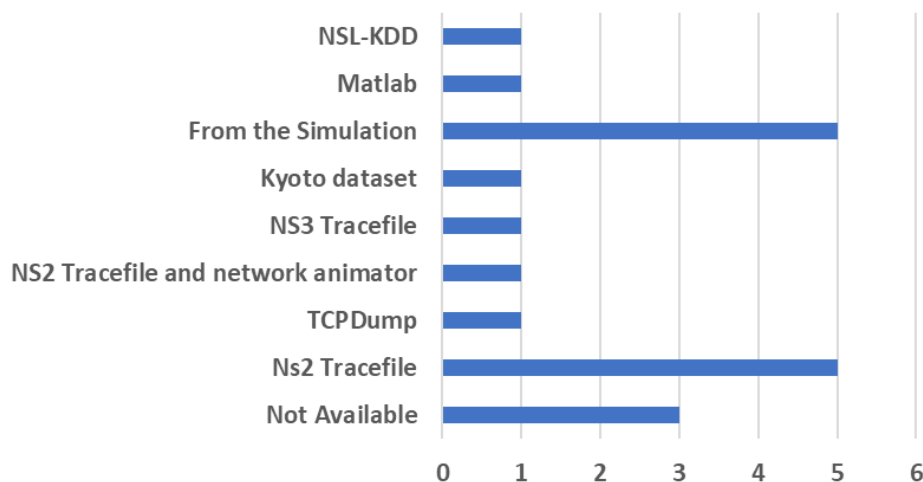


Figure 3.5: Origin of the Datasets Used in Intelligent IDSs Related Works

Figure 3.6 shows which ML algorithms were used in the studies. The most common was Neural Networks (NN), followed by SVM. Some of the studies also combine more than one ML algorithm.

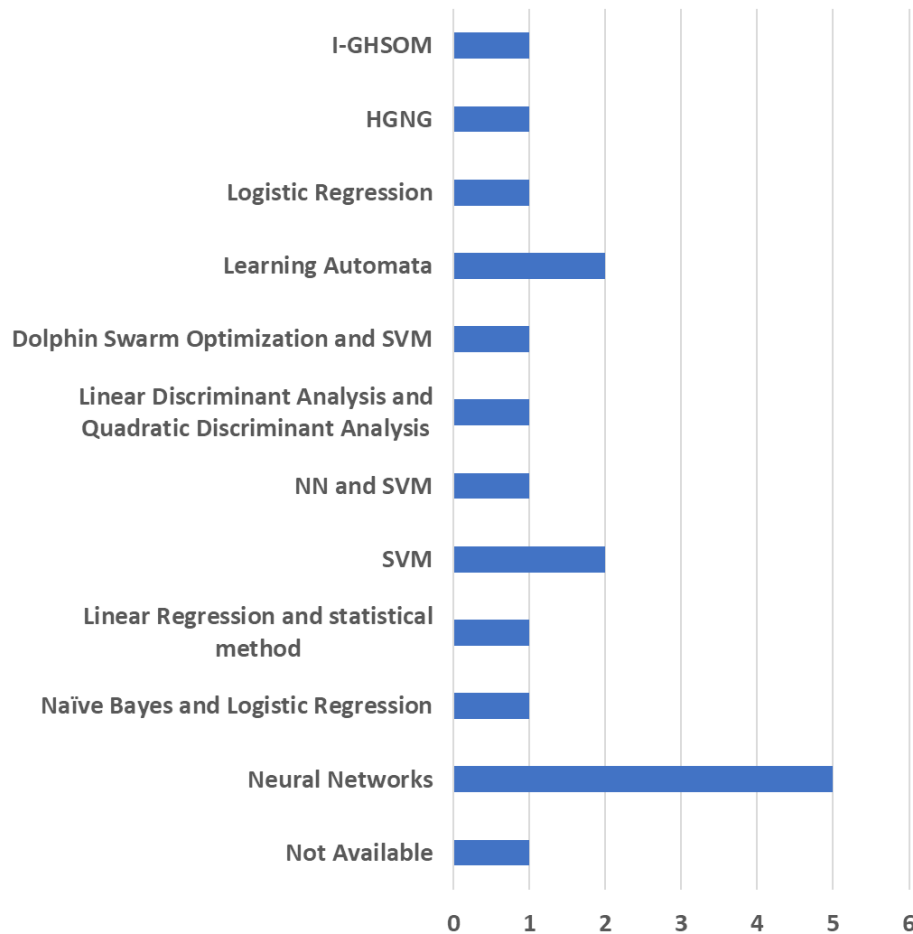


Figure 3.6: Types of Machine Learning Algorithms Used in Intelligent IDSs Related Works

## 3.2 Synthesis of Collected Data in the SLR

This section presents the summary of the collected works using the SLR methodology. It describes the major contributions, used tools, and strategies in the works found, providing more clear insight into the path followed in other research works.

**Misra et al. [75]:** In this work, the authors propose a Learning Automata (LA) based solution for IDSs in VANETs. The proposed model is privacy conscious, assigning a dynamic ID to each vehicle, making it untraceable.

In the proposed solution, a VANET management system is built. Each route has a base station, and all vehicles are equipped with transmitting devices required to communicate with the base stations. Each attacker

will create malicious packets to divert traffic.

Both the attackers and the system have a budget-based system. The bigger the budget for the attacker, the more packets it will be able to generate. For the IDS, the budget increases the sampling rate of VANET packets.

The attackers are detected using only their dynamically attributed IDs. If only un-allocated IDs are attributed and if more than one vehicle possesses the same ID, one of them is malicious. The learning automata is used to attribute the budget for under attack grids and re-calculating the sampling rate.

The model evaluation is performed by simulation, using their own simulator. Several simulations are done varying the number of attackers and the attacker and the IDS budget. The results vary from 40 to almost 100% of malicious packets caught depending on the system budget, the number of attackers, and the number of vehicles. In the proposed solution, the IDS is in each of the base stations and seems to have full visibility of the network.

**Tian et al. [76]:** Presents an IDS for VANETs based on the BUSNet. It is a virtual mobile backbone infrastructure constructed using public buses. In this solution, the buses act as cluster-heads, gathering the data packets transmitted by all vehicles and transmitting them to the access points along the roadsides. Then, this information is classified using a NN based algorithm and used to detect DoS attacks.

The presented solution is tested through simulation using the ns-2 network simulator. The authors do not indicate how the traffic is modeled or which traffic simulator is used. In the simulation, 50 vehicles transmit data with a Constant Bit Rate (CBR) and a packet size of 512 bytes. The sending period is of 4 seconds. The attacks are performed by two of the vehicles that transmit data with smaller time intervals, 0.01 seconds, at 4 distinct time points during 10 seconds each attack. The authors define a threshold value from which they consider an attack. This value is varied from 0.05 to 0.7, making the results vary. The optimal value for the threshold is 0.2.

**Kumar et al. [77]:** This work proposes an IDS based on trust-aware collaborative learning automata. Each vehicle has a data collection, detection, and alert generation module operated by automata. These modules are used in conjunction to collect information from the data sent between vehicles, according to their position and movement. Then, it is processed to detect attacks and generate alerts.

The solution is tested through simulation using Vanet MobiSim, with a total number of 500 vehicles with speeds between 20 and 50 km/h. Unfortunately, the authors do not indicate which network simulator is used. The authors then vary the number and speed of the nodes to evaluate their solution. The detection rate varies from 82% to 95%, depending on the number of nodes and their speed. The results have a more accentuated descent with the speed increase. Finally, the solution is compared with similar works.

**Liu et al. [78]:** The authors propose applying data mining methodology to detect known attacks and discover other unknown attacks in VANETs. The presented solution has three main contributions: a

decentralized vehicle network with scalable communication and data available about the network, using two data mining models to show feasibility for an IDS in VANETs and finding new patterns of unknown intrusions.

In the proposed system, the network is divided into a cell grid. Each cell has a transmission tower that enables communication with other cells and the Internet. Each one will run its own data mining models and rules, detecting new attacks. Thus, this allows the IDS to create new rules to be transmitted for each subnetwork. The data exchanged in the network is collected by both vehicles and the tower cell. The authors apply Naive Bayes and Logistic Regression classifiers to the collected data.

The authors first test the network performance of their method using simulation. First, SUMO is used to generate a mobility trace file. The trace file is fed into ns-2 to simulate the wireless network. This scenario comprises 150 vehicles that make random turning decisions at intersections, follow the speed limits (from 5 to 20 m/s), and randomly placed traffic lights.

The IDS was tested by loading 5 vehicles with Linux running several network applications. Then, TCPdump was used for 9 months to build the dataset. 4 attack categories with 39 attack types were recorded. These are then classified using WEKA. The evaluation of the models was made using the metrics recall, F-measure, and Matthews Correlation Coefficient (MCC).

**Mejri et al.[79]:** The proposed solution in comprises a detection algorithm based on a statistical method, linear regression, and watchdog to, in a passive way, be able to detect greedy behavior in the MAC layer. The proposed solution uses a watchdog to monitor the correlation of access times of active nodes. The algorithm considers the network to be under attack if the correlation coefficient is not close to 1, or if the correlation coefficient is close to 1, and the slope of the linear regression is not close to 1. The implemented software monitors the following metrics: duration between two successive transmissions, transmission time, and connection attempts of a node.

The performance evaluation of the solutions is made using the trace file generated by SUMO directly in ns-3. The scenario used in SUMO is based on a real city map with signs and traffic lights. Firstly the network is simulated with normal behaved nodes to confirm the application of the linear regression method. Then, there are injected greedy nodes, one by one, until a total of four.

The solution can detect the greedy behavior in 1.3, 1.9, 3.1, and 7.9 seconds for 1, 2, 3, and 4 nodes, respectively.

**Alheeti et al. [80]:** The solution proposed by these authors is an ANN-based misuse and anomaly IDS to detect DoS attacks.

To design the solution, firstly, the authors generated a mobility scenario using SUMO. The files generated by SUMO were converted using MOVE to be recognizable by ns-2. Finally, ns-2 was used to simulate communications between the vehicles. The authors used the Manhattan urban mobility to create the mobility and traffic scenario.

The simulation scenario is comprised of 30 vehicles and 6 [RSUs](#) and runs for 250 seconds. The vehicles run a [CBR](#) application that sends [UDP](#) packets. Only one of the vehicles is malicious. It will drop the packets instead of forwarding them.

The designed solution was able to classify normal and abnormal behaviors with 85.02% and 98.45% accuracy, respectively.

**Alheeti et al. [81]:** Authors propose the use of a Proportional Overlapping Scores ([POS](#)) method to reduce the number of features that are extracted from the trace file. These are used to train an [ANN](#) for classification.

The first step of the solution, the generation of the mobility scenario and trace file, is similar to the one presented in [\[80\]](#).

The generated [ns-2](#) trace files were then used in their [POS](#) algorithms to extract the more relevant features. Finally, the data from the dataset is then fuzzified to avoid classification problems.

The [IDS](#) designed in this solution uses Feed-Forward Neural Network ([FFNN](#)) to classify the dataset. 60000 dataset records were used, divided into training (50%), testing (25%), and validation(25%). This dataset contains both normal and malicious behavior. The malicious behavior crafted was a Black Hole attack, in which the malicious vehicle will drop all the received packets instead of forwarding them.

Finally, the [IDS](#) was tested using both anomaly and misuse detection. For anomaly detection, the normal and abnormal behavior results were 99.87% and 99.72%, respectively. In misuse detection, it obtained a classification of 99.89% for the normal behavior and 99.80% for the abnormal.

**Sedjelmaci et al. [82]:** Authors propose a cluster-based [IDS](#) that aims to protect the network against selective forwarding, black hole, wormhole, packet duplication, resource exhaustion, and Sybil attack. The proposed approach applies several detection agents that run at three levels - Cluster member, cluster-head, and [RSU](#).

The vehicles are grouped in clusters according to their velocities. The parameters used to select the cluster heads are cluster connectivity and assuring security. Connectivity within a cluster is improved by introducing a social behavior.

The [IDS](#) architecture is composed of two main detection systems and a decision system. The detection systems are the Local [IDS](#) and Global [IDS](#), which run at cluster members and cluster heads. The decision system is called Global Decision System and runs at the [RSUs](#), which allows the system to detect attacks at different levels and monitor the multiple entities. Also, each level can execute different algorithms and detection techniques, evaluating different features. The Global Decision System will receive the aggregate reputation of each vehicle forwarded by the cluster head and computes their trust level.

The solution was evaluated using [ns-3](#) as the network simulator and [SUMO](#) to simulate the mobility of

the vehicles. The IDS evaluation was made in terms of DR, FPR, and detection time. The several attacks were tested with a different number of vehicles. The number of attackers is fixed at 45% of the total vehicles. The results presented vary from 92% to 100%, depending on the number of vehicles and the attack tested.

**Alheeti et al. [83]:** A hierarchical intelligent IDS to secure communication for self-driving and semi self-driving cars is proposed. The authors use the Kyoto dataset and apply the POS algorithm to decrease the number of features. Then, this data is classified using Back Propagation Neural Network (BPNN).

The designed solution has five phases: preprocessing, feature selection, fuzzification, and training and testing.

The IDS is tested using a dataset of 60000 records extracted from the Kyoto dataset. This, is divided into three subsets: test (25%), validation(25%) and training(50%). The results obtained are 99.23% accuracy for normal behavior and 99.05% for abnormal.

**Alheeti et al. [84]:** In this work, an FFNN and SVM-based IDS is proposed. Also, a systematic response is proposed to protect vehicles when malicious behavior is detected.

The IDS is trained using a dataset built from SUMO and ns-2, using the features from the trace file. These features are reduced from 21 to 15 using the POS algorithm.

The dataset contains 30000 records that define normal and malicious behavior. This dataset is fuzzified before being used for classification. The authors subdivide the dataset into validation, test, and training.

Grey hole attacks are generated by selecting malicious vehicles that will drop packets at random times. For the rushing attack, the Ad hoc On-Demand Distance Vector (AODV) protocol needed to be adapted.

The accuracy of the results obtained in the simulation were 99.93% for normal behavior and 99.64% for abnormal behavior using SVM. Using FFNN, the results were 99.82% and 98.86% accuracy for normal and abnormal behavior, respectively.

**Wahab et al. [85]:** In this work, an intelligent IDS is proposed. It includes a cooperative monitor, able to collect messages exchanged by vehicles, and uses SVM in an online and incremental fashion to classify the vehicles. The protocol overhead is reduced by decreasing the training dataset. The decrease is done by restricting the data collection storage and analysis to only a set of specialized nodes and migrating a few tuples from one detection iteration to another. Also, a propagation algorithm is proposed that enables the dissemination of only the final decisions among clusters.

The data is collected by all cluster members that are designated as watchdogs. These will continuously monitor and analyze the MultiPoint Relay (MPR) vehicles that are serving them, detecting packet drops. Then, all the watchdogs in each cluster share their collected evidence. Afterward, each watchdog classifies the collected data. To do so, they use their own collected data as the test dataset and the observations of the other watchdogs as the training dataset.



The proposed solution was tested by simulation. Matlab Gilat has been used to implement the network-related algorithms and VanetMobiSim to simulate the road traffic. The simulations are made with several vehicles varying from a total of 100 to 500. The number of malicious vehicles varies from 10% to 50%. The solution has a detection rate of 98.1%.

**Alheeti et al. [86]:** Authors propose an intelligent IDS to protect against attacks, mainly DoS and Black Hole attacks, using Linear and Quadratic Discriminant Analysis.

Firstly, malicious behaviors are simulated. To generate the DoS attack, the authors modify the AODV protocol. In this case, the DoS attacks cause dropped router packets.

Then, the mobility and traffic scenarios were created. SUMO and MOVE were used to generate a realistic environment of malicious and normal behaviors. Additionally, ns-2 was used to simulate communications. The dataset is extracted from the ns-2 trace file, and all the 21 features are maintained. Before being used for the test and train, the data is fuzzified.

The obtained detection rate results are 86.44% for the Linear Discriminant Analysis and 85.67% for the Quadratic Discriminant Analysis.

**Sharma et al. [87]:** The Authors propose a new method for the selection of a stable Cluster Head, using Hybrid Fuzzy Multi-criteria. Then, a ML-based IDS using SVM is used to detect malicious behavior. The SVM-based IDS detection capabilities are improved by using a Dolphin Swarm Algorithm. This algorithm uses the dolphin swarm behavior of hunting and preying to detect and isolate malicious nodes in the network.

The proposed scheme is tested using simulation. NetSim and Matlab are used as the network simulators and SUMO for the traffic. In the simulation, several node densities are tested from 50 to 300, and a maximum of 45% of the vehicles are attackers.

The detection rate of the proposed IDS varies depending on the number of vehicles and is more than 98% for packet drop and selective forwarding, and wormhole attack.

**Nie et al. [88]:** In [88], an anomaly detection algorithm is proposed using the network traffic estimation made using the Spatio-temporal feature of the network traffic. The convolutional NN is used to extract features of the traffic matrix.

To detect the anomalies, first, the traffic is estimated based on the convolutional network. The anomaly detection is done based on a threshold identification approach.

There were 3000 tests carried out, and the results indicate a high True Positive Rate (TPR) mainly compared with previous works.

**Tan et al. [8]:** The main goal of this work is to propose a certificateless authentication scheme with Chinese remainder theory for efficient group key distribution. However, as several anomaly messages need to be authenticated during a relatively short period, there is the possibility of DoS attacks. So, an unsupervised

anomaly detection scheme is proposed, which applies time warping for distance measurement. In this scheme, vehicles need to maintain communication with the RSUs as they perform part of the work.

The built IDS is a multilevel hierarchy in which the clusters at one level are joined as clusters at the next level.

The proposed scheme is tested experimentally using Python and the pypbc library. The authors do not indicate any network or traffic simulator.

**Zhang et al. [89]:** Authors propose a privacy-preserving ML-based collaborative IDS. With that goal, a collaborative IDS architecture is proposed, enabling information and knowledge sharing. Then, an alternating direction method of multipliers algorithm is used to capture the distributed nature of the network and construct a collaborative learning algorithm over a VANET on a regularized empirical risk minimization algorithm. The privacy of collaborative learning is achieved by using the dual variable perturbation before minimizing the augmented Lagrange function. The goal of the classifier is to detect if the network is under attack using logistic regression.

The proposal is tested using the well-known NSL-KDD dataset evaluating the impact of the VANET size and topology.

**Ayoob et al. [90]:** In this work, authors propose using an IDS with a Hierarchical Growing Gas Network (HGNG) based classifier. Also, a semi-cooperative feature extraction method is used to collect the current location information, the location features, and the historical information.

The IDS is trained in a non-attack situation so the IDS can detect anomalies in the VANET. Each vehicle calculates measurements such as average traffic and location information.

Simulation is used to test the designed IDS. SUMO is used as the traffic simulator and ns-2 as the network simulator. The evaluation is done by inserting 50% of malicious vehicles and verify the network changes.

**Liang et al. [91]:** The authors propose a feature extraction algorithm and a classifier based on an Improved Growing Hierarchical Self-Organized Map (I-GHSOM). The proposed algorithm extracts two key features: the differences in traffic flow and position.

The proposed IDS consists of three modules: feature extraction, classifier, and response. The feature extraction module quickly translates the measurements in the messages to features. The traffic flow extracted is the difference in flows between two adjacent vehicles. The difference in position is the difference between the claimed and detected position.

The classifier module has been trained and can check if there are any deviations in the messages according to the features extracted. Finally, the response module takes action to assure the security of the network.

The performance of the scheme is evaluated using simulation. [ns-2](#) is used to simulate network communications and [SUMO](#) to simulate the mobility of the vehicles. The rate of rogue vehicles is varied from 10% to 40%. The [IDS](#) has a better performance with a lower number of vehicles. The [FPR](#) increases with the increment in rogue vehicle rate, and the [TPR](#) decreases. The values for the [TPR](#) vary from a little over 86.5% to 98%. The [FPR](#) is between 0.4% and less than 1.2%.

The data obtained from the [SLR](#) works is summarized in Table 3.3. The information in the table presents the following characteristics of each solution: network simulator, traffic simulator, attack targeted, [IDS](#) type, detection type, machine learning algorithm, source of the dataset, and the placement of the [IDS](#) in the network.

Table 3.3: Data Extracted from Studies found in the SLR

Paper	Net Sim	Traffic Sim	Attacks	IDS Type	Detection Type	ML	Dataset	Placement
[75]	Own	Own	Malicious packets	Hierarchical	Anomaly	Learning Automata	From Simulation	Base Station
[76]	NS2	N.A.	DoS	Hierarchical	Anomaly	Neural Networks	NS2 Trace file	Access Points
[77]	N.A.	VANET Mo-bisim	Abnormal behaviors	Colaborative	Anomaly	Learning Automata	From Simulation	Vehicles
[78]	NS3	SUMO	DoS, R2L, U2R, Probing	Hierarchical	Anomaly	Naive and Bayes Logistic Regression	TCPdump	Each cell and vehicle
[79]	NS3	SUMO	Greedy Behavior	N.A.	Watchdog	Linear Regression	From simulation	N.A.
[80]	NS2	SUMO and MOVE	DoS	N.A.	Anomaly and Misuse	Neural Network	NS2 Trace file	Any Node
[81]	NS2	SUMO	Black Hole	N.A.	Anomaly and Misuse	Neural Network	NS2 Trace file and Animator	N.A.
[82]	NS3	SUMO	Selective Forwarding, Black Hole, Packet duplication, Resource Exhaustion and Sybil attack	Hierarchical	Rule Based and Anomaly	SVM	NS3 Trace file	Vehicles and RSUs
[83]	—	—	DoS	Hierarchical	Misuse and Anomaly	Neural Networks	Kyoto Dataset	N.A.
[92]	N.A.	N.A.	Malicious behavior	N.A.	Watchdog	Bayesian Filter	N.A.	Every Node
[84]	NS2	SUMO and MOVE	Grey Hole and Rushing	N.A.	Anomaly	Neural Networks and SVM	NS2 Trace file	N.A.
[85]	Matlab	VANET Mo-bisim	packet dropping	Hierarchical	Watchdog and Anomaly	SVML	From Simulation	Vehicles
[86]	NS2	SUMO and MOVE	DoS and Black Hole	Standalone	Anomaly	Linear and Quadratic Discriminant Analysis	NS2 Trace file	Vehicles
[87]	NetSim and Matlab	SUMO	Wormhole, Selective Forwarding, Packet Drop	Hierarchical	Anomaly	SVM	NS2 Trace file	Vehicles
[88]	N.A.	N.A.	Traffic Anomalies	N.A.	Anomaly	Neural Networks	N.A.	N.A.
[8]	N.A.	N.A.	DoS	Hierarchical	N.A.	N.A.	N.A.	N.A.
[89]	-	-	Network Anomalies	Hierarchical	Anomaly	Logistic Regression	NSL-KDD	Vehicles
[90]	NS2	SUMO	Network Anomalies	Hierarchical	N.A.	HGNG	From Simulation	Vehicles
[91]	NS2	SUMO	Network Anomalies	N.A.	Anomaly	I-GHSOM	From Simulation	Vehicles

N.A.: Not Available

### 3.3 Evolution of the ML-Based Research Work

The SLR only covered the literature until 2019. Nonetheless, the literature has been constantly accompanied, following the evolution of the new research work. So, the works found in the literature after 2019 are also added to the works described in this section. These, however, were not obtained using the SLR methodology because the goal was to follow the trends and path of new research on the literature, not needing such systematic and deep research as previously done in the SLR.

**Ghaleb et al. [93]:** Authors in propose a misbehavior-aware on-demand collaborative IDS based on the concept of distributed ensemble learning. So, each individual vehicle uses an Random Forest (RF) algorithm to train a local IDS. These are then shared on-demand with the vehicles in their vicinity to reduce the communication overhead. The vehicles that received the trained classifiers test them using a local test dataset. The results of the testing the classification is used as a trustworthiness factor to rank the received classifiers. The classifiers that deviate too much are excluded from the set of collaborators. The classifiers received from the multiple neighbors are aggregated using a weighted voting scheme. The mobility of the vehicles is simulated using SUMO, using five traffic scenarios with different vehicle densities, random vehicle types, speed, and behavior. The authors used the known NSL-KDD dataset to represent the vehicle network traffic and used three ML algorithms to evaluate the data, RF, XGBoost, and SVM. The IDS performed quite well, mainly using the RF algorithm, with F1 scores ranging from 98% to 99% depending on the number of misbehaving vehicles.

**Kosmanos et al. [94]:** The ML-based IDS that is proposed targets spoofing attacks using a probabilistic cross-layer approach in a VANET comprised of Electric Vehicles. If an attack is detected, the attackers are excluded from the Dynamic Wireless Charging mechanism. One of the contributions of the papers is the introduction of a novel metric used to separate features for the ML algorithms, which is named Position Verification using Relative Speed. It is based on the relative speed that is estimated through the interchanged signals in the PHY layers. The introduction of the new metric increased the performance of the probabilistic IDS by 6%. The authors designed their simulations and attacks using SUMO and OMNET++/VEINS simulators. The data created was evaluated using k-Nearest Neighbor and RF. The performance of both algorithms using the new metric for both was very similar with 91.3% accuracy.

**Gad et al. [95]:** propose an ML-based IDS for VANETs based on the ToN-IoT [96]. This dataset is an updated version of the NSL-KDD dataset, containing the most updated attacks. The authors use the SMOTE technique to fix the class unbalance and then compare the performance of the following algorithms in attack detection: Logistic Regression, Naive Bayes, Decision Tree, SVM, k-Nearest Neighbor, RF, and XGBoost. The dataset is divided using 70% for the training and validation of the algorithm and the rest for testing its performance. The results show that XGBoost has the best performance either in binary class and multi-class classification problems.

**Alsarhan et al. [97]:** the authors present a SVM-based IDS for VANETs, using an enhanced penalty function for reinforcing the regularization of the classifier and comparing three different ML algorithms for optimization. The test and training are made using the NSL-KDD dataset by dividing it into ten groups. One is used for the test, and the remaining 9 are used for training. The results show that SVM performs better when optimized using the Genetic Algorithm.

**Raja et al. [98]:** SP-CIDS is a Secure, and Private-Collaborative IDS proposed to detect network attacks and mitigate security concerns. It uses a distributed ML model based on the Alternating Direction Method of Multipliers. This algorithm leverages the potential of vehicle-to-vehicle collaboration in the learning process to improve the storage efficiency and accuracy, and scalability of the IDS. Nonetheless, the methodology used creates privacy concerns as the CIDS may act as a malicious system with access to the learning process's intermediate stages. So, the authors use a differential privacy technique to address the data privacy risk. The authors evaluate multiple ML algorithms in the IDS, including logistic regression, Naive Bayes, and ensemble classifiers. The authors implement the proposed SP-CIDS in simulation using ns-2. The classifier and differential privacy techniques are implemented in python and tested using the NSL-KDD dataset. The results show that the proposed IDS is very efficient in detecting attacks, mainly when using the ensemble learning technique. However, the results presented show only the total accuracy and not the accuracy per type of attack thus, not being clear of how the methodology behaves for each attack.

The data obtained from the studies collected during the follow-up on the literature evolution is summarized in Table 3.4.

Table 3.4: Data Extracted from Studies Follow-Up Research

Paper	Net Sim	Traffic Sim	Attacks	IDS Type	Detection Type	ML	Dataset	Placement
[93]	N.A	SUMO	Malicious behavior	Colaborative	Anomaly	Random Forrest, XGBboost and SVM	NSL-KDD	Vehicles
[94]	VEINS	SUMO	Spoofing	Standalone	Anomaly	Random Forrest, and k-Nearest Neighbor	From Simulation	Mobile Energy Disseminators and Static Charging Stations
[95]	N.A.	N.A.	Network Anomalies	N.A.	Anomaly	Logistic Regression, Naive Bayes, Decision Tree, SVM, k-Nearest Neighbor, Random Forest and XGBoost	ToN-IoT	N.A.
[97]	N.A	N.A.	Network Anomalies	N.A.	Anomaly	SVM	NSL-KDD	N.A.
[98]	NS2	N.A.	Network Anomalies	Colaborative	Anomaly	Logistic Regression, Naive Bayes, Ensemble Classifiers	NSL-KDD	Every Node

N.A.: Not Available

### 3.4 Analysis of Experimental Results

The most common approach for an IDS for VANETs uses the tools shown in Table 3.5. That solution comprises of a hybrid detection that performs misuse and anomaly detection. The simulation is performed using ns-2 as the network simulator and SUMO to simulate the vehicle's mobility. In most of the solutions, the dataset is obtained directly from the network simulator trace file.

Table 3.5: Most Common Tools Used in Intelligent IDS Solutions Found in the SLR

Network Simulator	NS-2
Traffic Simulator	SUMO
Dataset	Net Simulator Trace file
ML algorithm	NN
IDS Type	Hierarchical

Unfortunately, a significant number of studies did not specify which network and traffic simulator were used. The same happened for the dataset used to train and test the IDS.

Most of the collected studies use their own datasets, either from collected events from the simulation or

directly from the network simulator trace file. The ones that use more reliable datasets choose the well-known NSL-KDD [99] and Kyoto [73] datasets.

Both mentioned datasets are obtained from real network traffic. NSL-KDD is an evolution of the KDD CUP 99 Dataset [100], which is based on a dataset composed of about 4 gigabytes of compressed raw TCPDump data consisting of 7 weeks of network traffic. The original KDD dataset contains 41 features and is labeled as normal or attack messages. The collected messages can be categorized into the following groups: DoS, User to Root Attack, Remote to Local Attack, and Probing Attack.

The Kyoto dataset has more than three years of real traffic data collected from honeypots located in 5 different networks inside the Kyoto University. It consists of 14 statistical features derived from the KDD CUP 99 dataset with additional features for further evaluation and analysis of IDSs.

These are publicly available datasets with a high reputation and are used by many works. However, they are not obtained from VANET networks and thus may produce biased results.

The studies that use their own datasets do not make them publicly available, compromising the validity of their results. The unavailability of the datasets, combined with the lack of explanation on how the datasets are constructed and the clear description the process of building the attacks (ratio between normal and abnormal messages, etc.), complicates the verification of the results. Moreover, the great majority of the presented results indicate a very high value of the detection rate, which may indicate the overfitting of the model. Furthermore, most of the studies do not present the configuration parameters used in the algorithms.

The more recent work (2019-2021) seems to increasingly use publicly available datasets. These, however, choose the already mentioned NSL-KDD or a new dataset called ToN-IoT. Unfortunately, the latter still is not obtained from VANETs but from Internet of Things (IoT) networks, compromising the results of the works presented.



# Chapter 4

## Securing Communications

**VANET**s have a specific set of characteristics that makes them inherently vulnerable, creating opportunities for a wide range of attackers. Moreover, their need for secure but low latency communications requires a set of particular security requirements. Thus a security framework able to protect the communications of all the diverse entities that constitute the network is needed.

**VANET** security is a key area with multiple research works that propose different techniques, from the most traditional **PKI** to some more new and complex ID or situation-based. However, the solutions found in the literature present some disadvantages and do not address the intrinsic broadcast needs of **VANET** communications. So, it was decided to design a secure model that could overcome some of these drawbacks by incorporating multiple technologies into a hybrid security model.

Vehicular Ad hoc Network Public Key Infrastructure and Attribute-Based Encryption with Identity Manager Hybrid (**VPKIbrID**) [14] is an application layer security model that provides a framework enabling **VANET** entities to exchange messages securely while maintaining their privacy. This security model uses a hybrid of a **PKI** with **PCs** and **ABE** encryption. It can guarantee that **VANET** communications can be made securely. The **ABE** cryptography enables broadcast scenarios in which an entity can encrypt a message to multiple targets without needing exchange keys.

The entities and interactions of the **VPKIbrID** security model are presented in Figure 4.1. It is built on top of the **PKI** model with **PCs**, with two new additional entities, **TA** and **IdM**. The **TA** enables the usage of **ABE**. It is responsible for generating **ABE Public Parameters** needed to encrypt data and the decryption keys based on the attributes of the entities. The **IdM** can be used as an authorization server that generates OAuth/OpenID connect like tokens. These can be used to access services or resources while protecting privacy. The token is signed by the **IdM** to prove its authenticity. It can be used, for example, to obtain **PCs** from the **PCA**. The **PCA** will be able to verify the token authenticity but not the identity of the owner. OpenId Connect is an established Internet standard and is used as a basis for the token.

VPKIbrID security model enables encryption to one or multiple targets. If the message destination is a single entity, VPKIbrid Public Key Infrastructure (VPKIbrID-PKI), more lightweight, may be used. If the goal is to disseminate a message to a group of entities that shares the same attributes, VPKIbrid Attribute Base Encryption (VPKIbrID-ABE) is the more indicated. The message construction is similar in both modes and similar to the one used by WAVE.

The following Sections describe the multiple communication modes, message formats, and the several protocols needed to communicate securely.

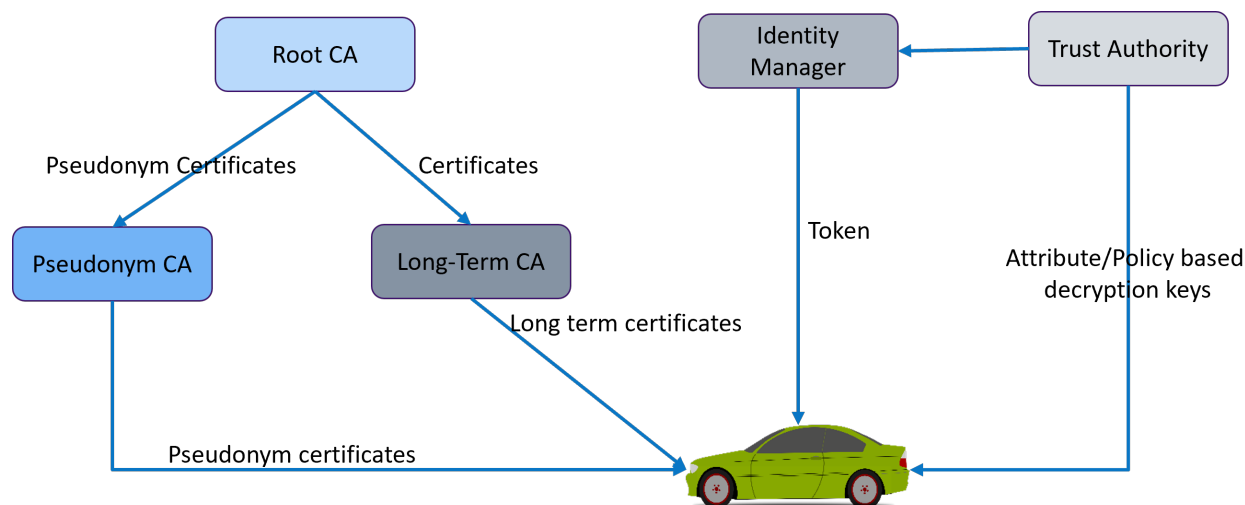


Figure 4.1: VPKIbrID Security Model Interactions (From [14])

## 4.1 VPKIbrID Message Format

VPKIbrID defines two different messages: the VPKIbrid Plain Message (VPKIbrID-PM) and the VPKIbrid Encrypted Message (VPKIbrID-EM). The first is the message in plain text that will be encrypted; the latter is the encrypted message that will be sent. Both are defined using the widely accepted format Abstract Syntax Notation One (ASN.1).

The VPKIbrID-PM definition is presented in Figure 4.2. It is composed of 7 fields: plain data (data), sender certificate (certificate), message creation UNIX timestamp (timestamp), IdM token (token), the algorithm used to sign the data (signatureAlgorithm), the key used to sign the data (signingKey) and, finally, the signature (signature). The sender certificate is also encrypted to protect anonymity.

The VPKIbrID-EM, presented in Figure 4.3, is composed of 6 fields: encrypted key (encKey), encrypted data (encData), data encryption algorithm (encAlgorithm), key encryption algorithm (encKeyAlgorithm), UNIX timestamp (timestamp), and message hash (hash).

```

VPKIbrID-PM := SEQUENCE {
    data OCTET STRING,
    certificate OCTET STRING,
    timestamp INTEGER,
    signatureAlgorithm PrintableString,
    signingKey OCTET STRING,
    signature OCTET STRING
}

```

Figure 4.2: [VPKIbrID-PM](#) format in [ASN.1](#)

The field `encData`, refers to the encrypted data and `encAlgorithm` to the algorithm used for its encryption. The same happens for `enc_key` and `encKeyAlgorithm`. The `encKey`, was used to create the `encData` from the [VPKIbrID-PM](#). And then, it was encrypted using the receiver public key. The hash field contains a hash obtained from the message. Its goal is to prevent simple modifications of the message.

```

VPKIbrID-EM := SEQUENCE {
    encKey OCTET STRING,
    encKeyAlgorithm OCTET STRING,
    encData PrintableString,
    encData OCTET STRING,
    encAlgorithm PrintableString,
    timestamp INTEGER,
    hash OCTET STRING
}

```

Figure 4.3: [VPKIbrID-EM](#) format in [ASN.1](#)

## 4.2 VPKIbrID Cipher Modes

[VANET](#)s are a very diverse environment where the entities may need to communicate in different modes. Furthermore, these may need to send messages to several targets (broadcast/multicast) or only one (unicast). Therefore, one goal of the designed [VPKIbrID](#) model was to enable the possibility to encrypt data depending on the communication type needed. [VPKIbrID-PKI](#) provides unicast communication using the capabilities of [PKI](#), and it can take advantage of [ABE](#) to provide an easy way to encrypt data for multiple targets simultaneously.

Figure 4.4 presents the block diagram for the encryption process of both modes [VPKIbrID-PKI](#) and [VPKIbrID-ABE](#). The encryption process is very similar in both methods, with the difference being in the "Receiver Pub Or ABE Key" block. This block may use a public key if the sender needs to perform a unicast

communication using [VPKIbrID-PKI](#) or use an [ABE](#) key for a broadcast communication using [VPKIbrID-ABE](#). Both methods are described in detail in the following Sections.

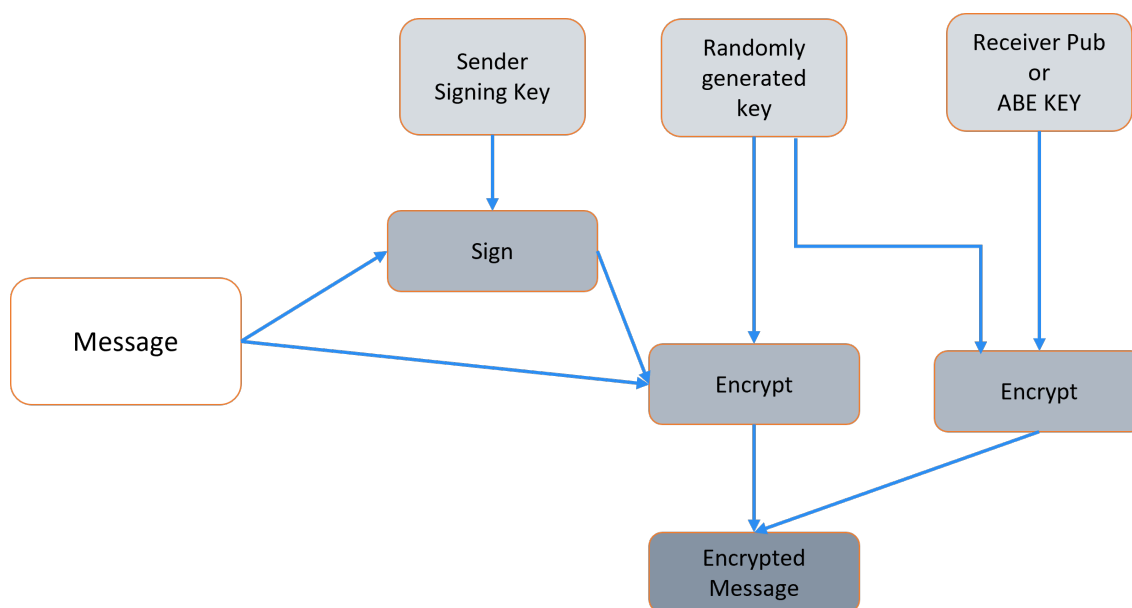


Figure 4.4: VPKIbrID-PKI and VPKIbrID-ABE Block Diagram

### 4.2.1 VPKIbrID Public Key Infrastructure

[VPKIbrID-PKI](#) is the encryption mode more indicated to be used in unicast communications, that is, any communication that only has one target, and the targets are known previously to the communication. It is based on the well-known, and more traditional, [PKI](#), a certificate-based cryptographic scheme. So, for the entities to be able to communicate securely, they need a pre-exchange of certificates. Therefore, all entities need to pre-load a [CA](#)-generated certificate with its respective private and public keys.

Figure 4.5 presents the interactions between the sender and receiver needed to secure communications using [VPKIbrID-PKI](#). **(1)** the sender generates a random symmetric key that will be used to encrypt the [VPKIbrID-PM](#); **(2)** then, the sender signs the data and builds the [VPKIbrID-PM](#), the signature can be done using a [PKI](#) or any symmetric algorithm such as a Message Authentication Code ([MAC](#)); **(3)** it encrypts the data and signature with the symmetric key, this key is faster and lighter than any [PKI](#) being optimal for this step; **(4)** then, the sender can encrypt the symmetric key using the public key of the receiver, using the slower encryption to encrypt only part of the data thus, increasing the efficiency; **(5)** the sender can now fill all the [VPKIbrID-EM](#) fields and send the message over the unsecured network securely.

The receiver can decrypt the message using the inverse process, also presented in Figure 4.5. So, it first **(6)** verifies the authenticity and expiration date of the certificate in the [VPKIbrID-EM](#). If valid, the receiver

can now **(7)** decrypt the symmetric key using its private key; **(8)** then it can decrypt the [VPKIbrID-EM](#) and obtain the [VPKIbrID-PM](#); And, finally, **(9)** it can verify the signature in the message. If correct, it means that the message was successfully transmitted.

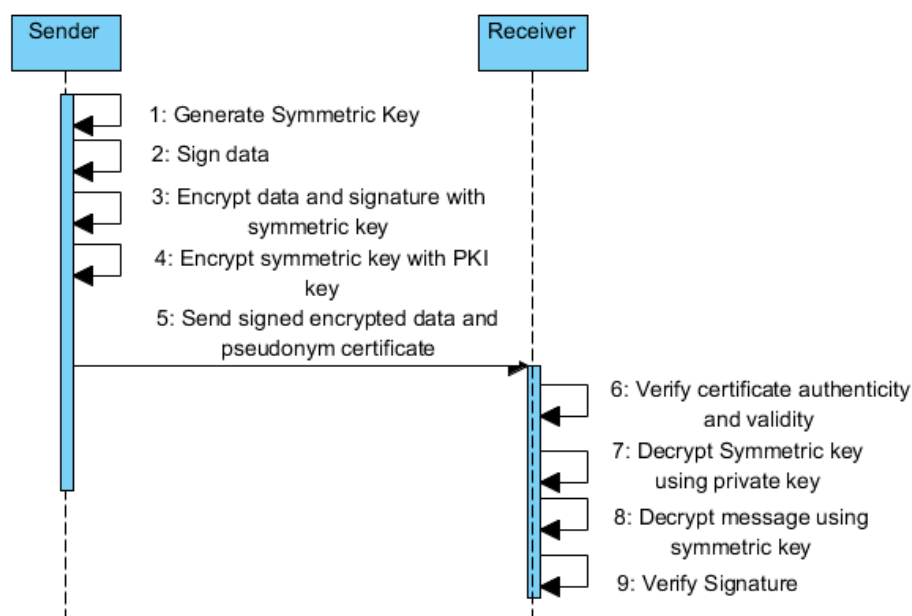


Figure 4.5: [VPKIbrID-PKI](#) Encryption and Decryption Sequence Diagram

## 4.2.2 VPKIbrID Attribute-Based Encryption

[VPKIbrID-ABE](#) mode uses the more complex [ABE](#). This mode has some drawbacks, including the [CPU](#) demand and the increased time in the encryption and decryption process. Also, it has only a few available implementations, is usually made by individuals, is less used and, thus, less efficient. However, it presents some advantages, the main and already mentioned is the capability of performing encryption in such a way that enables multiple targets to decrypt the message as long as they share the same attributes. It is optimal for use in broadcast/multicast scenarios where an entity needs to broadcast information to a restricted group without knowing their public keys.

Furthermore, as it uses attributes for its public key, it can encrypt messages for targets that do not already exist. For example, if a cloud-based service is used, the sender can encrypt a message for users with the role "Admin," even if no user has that role. That role can afterward be added to the system, and that user will immediately have access to the data. So, it provides an access control mechanism embedded in the encryption. Another possibility is also to define an expiration date in the attributes, from which no one will be able to decrypt the data.

The entities need to be pre-loaded with the **Public Parameters** to cipher and decipher the data. These

are generated by the **TA** and are needed for the cipher and decipher process. The **Public Parameters** can be sent through an insecure channel (only need to be signed) as all entities know them.

The encryption process, presented in Figure 4.6, is similar to the **VPKIbrID-PKI** mode. The only difference in the encryption of the symmetric key. Instead of being encrypted by the receiver public key, it will be encrypted with an **ABE** key. An example of attributes that can be used is **"UMINHO DI 2of2"**. It indicates that only entities that belong to the group **UMINHO** and are from **DI** can decipher the message. The **"2of2"** part of the rule means that the two attributes need to be fulfilled.

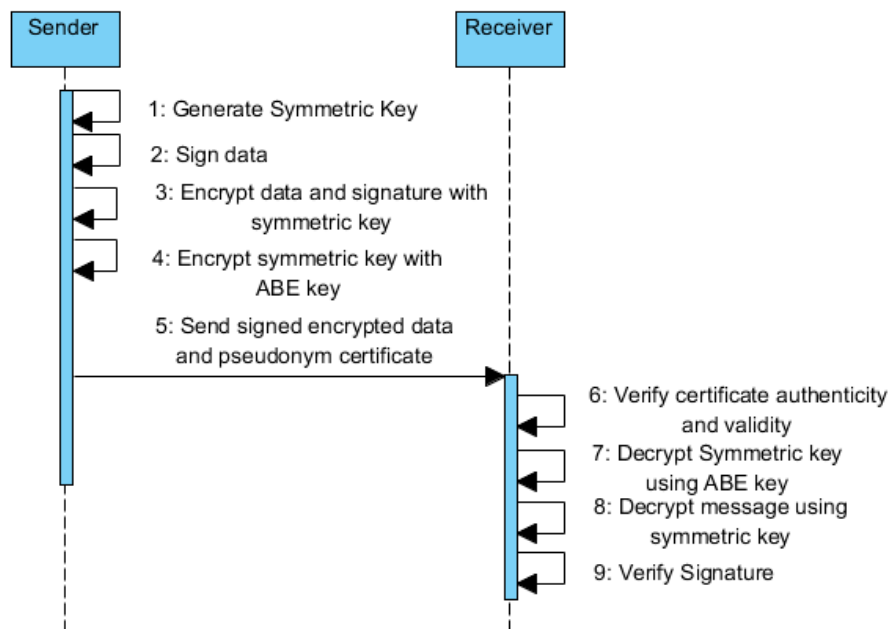


Figure 4.6: **VPKIbrID-ABE** Encryption and Decryption Sequence Diagram

### 4.3 VPKIbrID Interactions

The **VPKIbrID** security model comprises multiple entities with different roles, including the **TA**, Root **CA**, **PCA**, Long-Term **CA**, and **IdM**. To be able to communicate securely, any sender/receiver needs to obtain cryptographic material from them. However, the vehicle is assumed to be pre-loaded with the following material: Long-Term Certificates (**LTC**), **Public Parameters** generated by the **TA** and, the **TA** and **IdM** certificates. All the indicated security material has a long expiration date and thus can be pre-loaded during, for example, the yearly mandatory vehicle revision. Next are presented the interactions needed for the **VANET** entities to obtain the security material needed to create a secure communication channel.

### 4.3.1 Obtaining Identity Manager Token

The token is one of the fundamental parts of the [VPKIbrID](#). It can be used to access services or resources without leaking private information. The [VPKIbrID](#) token is very similar to Oauth, providing access control management (not actually authentication). Despite the [VPKIbrID](#) security model having the [LTC](#) for entities to prove their identity to each other, it leaks the true identity. So, this certificate should only be used to authenticate with trustworthy entities such as the [IdM](#) or the [CAs](#).

The token is a [JWT](#), similar to the one used by OpenID connect. It describes the resource it gives access to, has an expiration date, and is signed by the [IdM](#). Thus, any entity that receives it can automatically verify its validity without the need for any extra communication. However, it does not indicate the entity it has been generated for, protecting its privacy.

Any entity that wants to obtain a token can do so by requesting it from the [IdM](#). The request should include the [LTC](#) of the requester and what resources it wants to access (obtain the [ABE](#) keys, join a platooning, etc.). As the request is made through a unicast communication, in which only the [IdM](#) should access the data, it should be made using [VPKIbrID-PKI](#). The request should be encrypted using the [IdM](#) certificate pre-loaded along with the rest of the secure material.

So, to obtain the token, the requesting entity first **(1)** encrypts the request and its [LTC](#) certificate using the [PKI](#) mode. **(2)** Then, it sends the request to the [IdM](#). The [IdM](#) receives the token and proceeds to: **(3)** decrypt the request using its private key; **(4)** generate the token according to the specific permissions of each entity. It should clearly indicate which services the entity has access to and, if needed, which are its attributes; **(5)** then it signs the token with its public key, thus being easily verifiable by any other entity in the system, **(6)** encrypts the token with the requester public key and **(7)** sends it over the network. Upon receiving the token, the requester can **(8)** decrypt the token using its private key. The whole process can be seen in Figure 4.7.

The token should be kept securely and only be transmitted over a secure channel to trustable entities. It is impossible for the entity who receives the token to identify the sender, only the validity of the token and the resources it gives access to. So, if an attacker can intercept the token, it can access valuable resources. One possible way to try to mitigate this is to define low expiration dates.

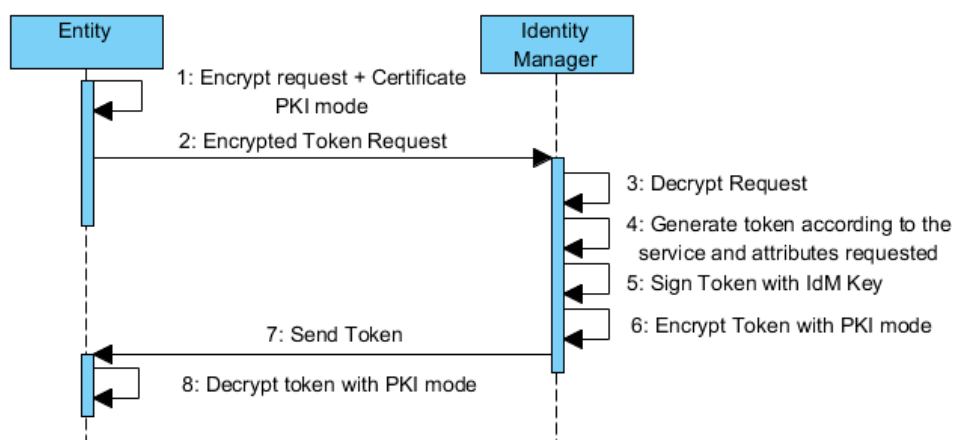


Figure 4.7: Request VPKIbrID Token From the IdM

### 4.3.2 Obtaining Pseudonym Certificates

Certificates are one of the fundamental components of a PKI. These are a widely accepted form for entities to identify securely and link a public key to identification. However, certificates have multiple parameters that allow the identification of their owner in cleartext. In, reality even the public key can allow the identification of an entity across the network. So, to prevent this from happening, the entities can use PCs. These are very similar to the LTCs but hide the real identities of their owner, protecting their privacy. Nevertheless, as an authentic and well-known entity signs these, it is possible to confirm their authenticity. Each entity can have multiple PCs that can be interchanged, preventing tracking any entity in the network.

An entity that wants to request a PC from the PCA must first obtain the IdM token, as described in Section 4.3.1. The token needs to indicate that the entity may access the PCs and which attributes it must contain. It also needs to have the PCA certificate and its LTC pre-loaded. Then the requester entity (1) encrypts the token and the request using VPKIbrID-PKI with the PCA public key, and (2) sends it to the PCA. Upon receiving the request, the PCA can (3) decrypt it with its private key and (4) verify the validity of the token. If valid, it (5) generates the PCs that will be (6) encrypted using the requester public key and (7) sends them over the network. The requester can then use VPKIbrID-PKI to (8) decrypt the symmetric key and (9) with it the encrypted certificates. The process is described in detail in Figure 4.8.

As previously mentioned, the VPKIbrID-PKI supports message signing using PKI or symmetric algorithms. In this case, both the token in the request and the PCs are signed; the entities may choose to use symmetric keys to sign message. It may be especially useful for entities with less computing power, helping to increase their performance.



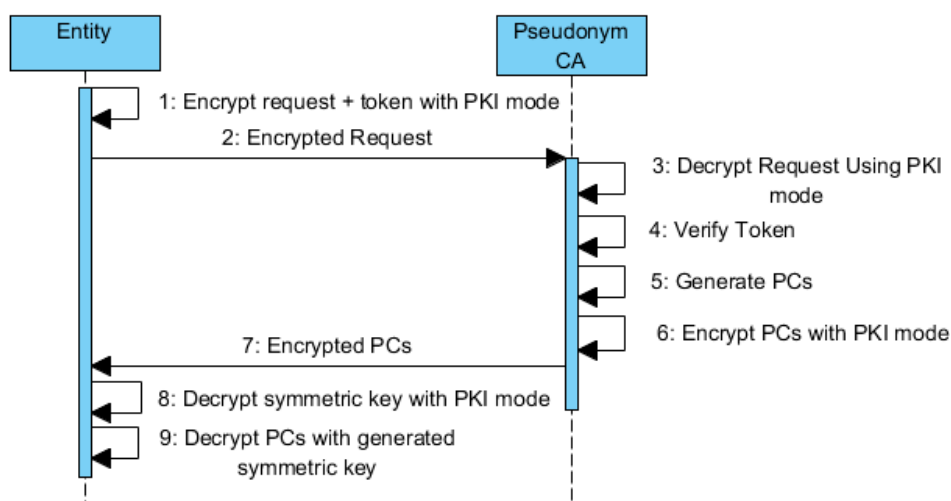


Figure 4.8: Request VPKIbrID PCs From the PCA

### 4.3.3 Obtaining Attribute Base Encryption Keys

Unlike the more traditional methods, ABE is a type of cryptography that uses attributes to encrypt the data. Still, this type of cryptography is less used and has little representation in the market. Nevertheless, ABE keys present some advantages when compared with the more traditional techniques. These allow an entity to encrypt data that multiple targets can decrypt. Additionally, ABE keys have an embedded access control mechanism that is very useful to secure messages sent to groups of entities that share the same attributes and can even be used to encrypt data that will only be decrypted in the future.

In this scheme, the sender can directly generate the encryption key using the intended attributes, only needing the **Public Parameters** generated by the TA. However, the decryption keys can only be generated by the TA, a trusted authority in the system that is considered secure. Then, the keys can be requested using a token from the IdM that proves the attributes of the requesting entity.

First, to request the ABE keys, the requester needs to obtain the IdM token from the IdM, as indicated in Section 4.3.1 and represented by steps 1-8 in Figure 4.9. Then, the (9) requester encrypts the token and a PCA using the VPKIbrID-PKI mode with the TA certificate and (10) sends it over the network. The IdM can then (11) decrypt the token and verify its authenticity, (12) get the attributes from the token, and (13) generate the decryption keys based on the attributes in the token. The TA can then (14) encrypt the token using VPKIbrID-PKI with the public key indicated in the PCA and (15) send it back to the requester. Finally, (16) the requester can decrypt the ABE key using its PCA. The complete process can be seen in Figure 4.9.

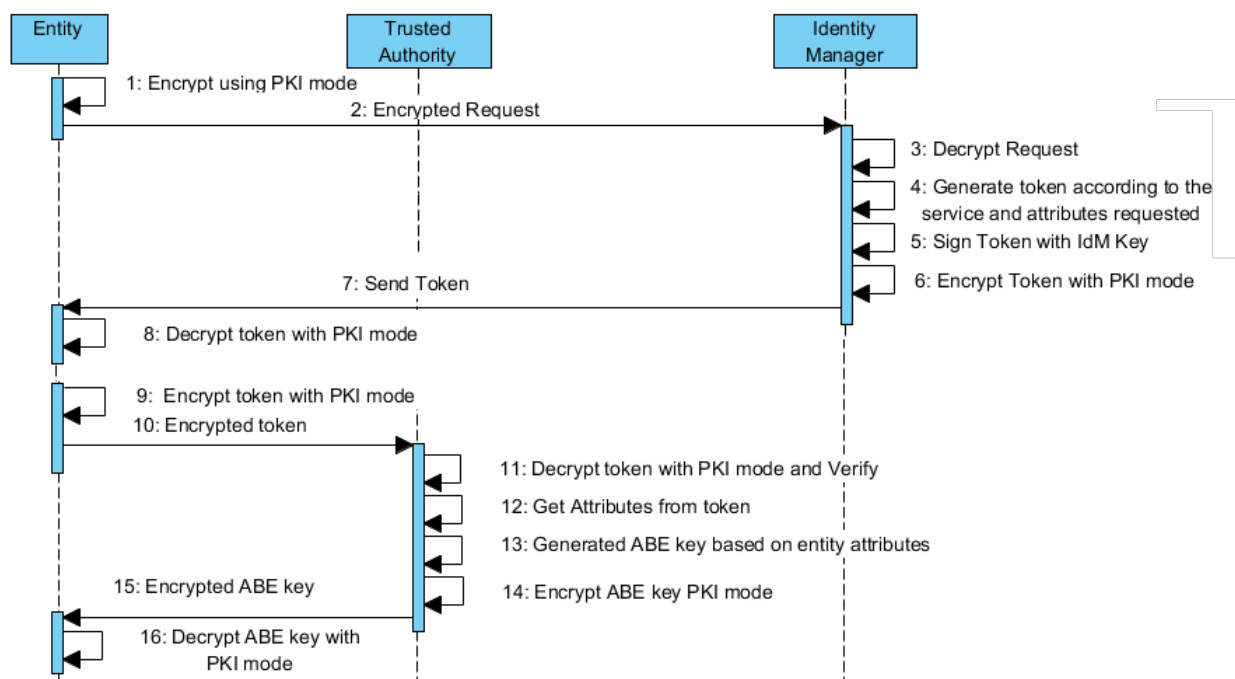


Figure 4.9: Request VPKIbrID ABE Keys From the TA

### 4.3.4 User Authentication

VANET security is needed at multiple levels. Depending on the application, different permissions can be given to the driver depending on its identity. In some more security constrained applications, such as platooning, where the driver of a vehicle can directly impact the behavior of numerous others, the driver needs to be identified, or, at least, be able to prove his permissions. The user authentication is mainly essential if the vehicle belongs to a Leader. The Leader is responsible for controlling all the maneuvers within a platooning, and it should have special permission given to a user that has the training to do so.

VPKIbrID provides tools to do so by using a combination of the LTC and the token. So, the driver can be given a secure storage device, such as a smartcard containing his LTC and corresponding keys. The LTC can then be used to easily obtain the user token from the IdM, which indicates the permissions that the user has. The user token should only give permissions for access to the vehicle or the needed application; these should not give access to fetch PCs or ABE keys. The IdM token can also be stored in the user smartcard for later use if the expiration date is enough. The token can be obtained offline, using the resource website, for example, and storing the token in the card. Alternatively, in the vehicle using a process similar to the one described in Section 4.3.1. The only difference is that the LTC used is the user instead of the one belonging to the vehicle.

Figure 4.10 shows how the user token can be obtained. **(1)** the user inserts the smart in the vehicle. The **(2)** vehicle requests the user credentials, which were used to secure the LTC. **(3)** the user inserts his credentials to allow access to his information. The device can then **(4)** verify the credentials and **(5)** asks

which service the token is intended for. The user **(6)** inserts the service identification, and the device can encrypt the request **(7)** and send it to the IdM **(8)**. The IdM can then **(9)** decrypt the certificate, verify its authenticity and validity **(10)** and generate and encrypt the token using the PKI mode **(11)**. The IdM sends it to the device, which **(12)** stores it securely in the smart card **(13)**. The user can finally remove the smart card **(14)** with its token inside.

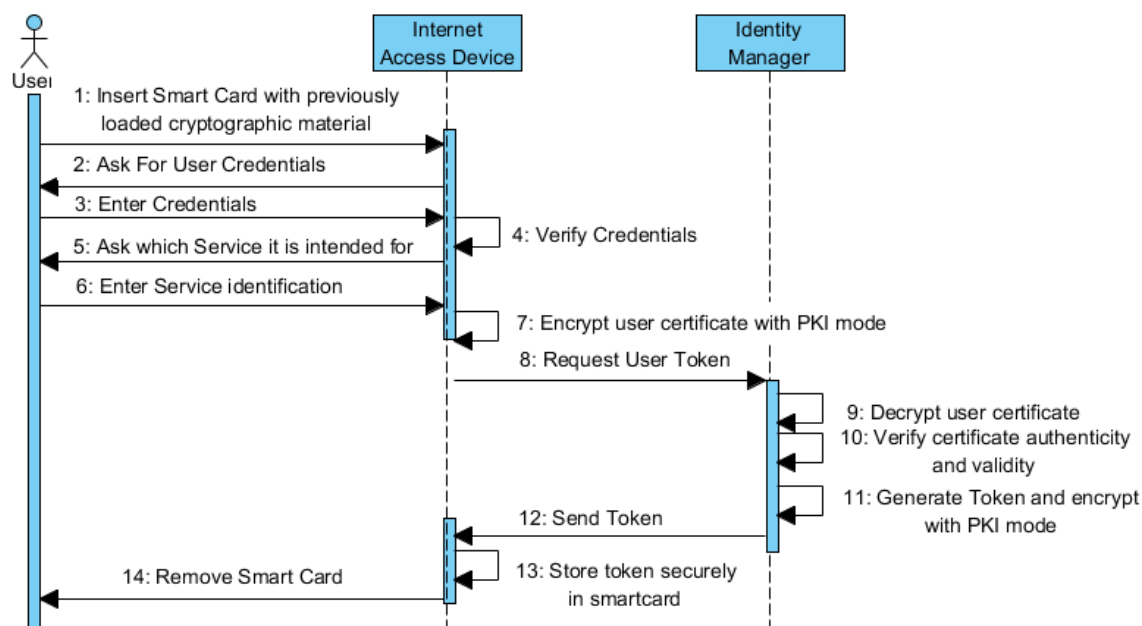


Figure 4.10: VPKIbrID User Authentication

The user token can be used to access the vehicle. Entering or driving a vehicle can be authorized through the token. When buying a vehicle, the owner should be given a token that authorizes him to do so. This token can be long-lived as it is not exchanged in the network. Also, it is only supplied to the vehicle after the authorized by the owner.

The user will have to enter his smartcard in the vehicle, which, in turn, will prompt for his credentials. After validation, the vehicle will have access to the token. It is signed by the IdM, allowing its verification. The complete process is presented in Figure 4.11.

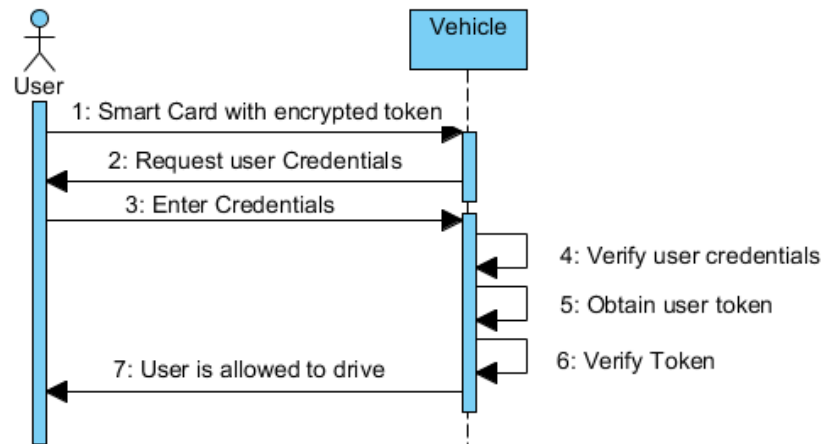


Figure 4.11: Using the VPKIbrID User Token to Access the Vehicle

## 4.4 Secure Platooning

Platooning is an [ITS](#) application that aims to improve traffic flow and highway capacity while increasing safety. It allows multiple vehicles to travel close together with constant speeds and gaps in a convoy manner. A Platoon may be composed of a diversity of vehicles, from trucks to normal automobiles. The vehicles that compose a Platoon may perform one of two roles, **Follower** or **Leader**. The **Leaders** usually are the vehicle in the front of the Platoon and control all the maneuvers and movements of the Platoon. The **Followers** are the rest of the vehicles and comply with the orders of the **Leader**.

It has complex security requirements, most related to its underlying network, the [VANET](#), and the nature of the application itself. Due to their complexity, some authors even suggest dividing the requirements into three layers, as indicated in Section 2.8.2: Communication Channel Security Requirements, System Layer Security Requirements, and Application Layer Security Requirements. Additionally, the intricacy of the application and its maneuvers facilitates attackers to influence the behavior the vehicle, compromise privacy of the drivers or cause attacks. Thus the importance of a secure channel with strong authentication to support the message exchange within the Platoon.

Communications between Platoon vehicles can happen in different ways. The vehicles may need to use broadcast/multicast communications when transmitting a beacon to make themselves known or unicast communication during most of the maneuvers, where most of the communications are between the followers and the leader.

[VPKIbrID \[14\]](#) is an application layer security model that provides mechanisms to facilitate broadcast/-multicast and unicast communications while creating a secure channel with strong authentication and maintain user privacy, as described in Section 4. In conjunction with its User Authentication Capabilities (Section 4.3.4),

the secure channel created by [VPKIbrID](#) can fulfill the platooning security requirements as described in Section [2.8.2](#).

The next Sections describe the multiple interactions between the Platooning entities to perform the maneuvers presented in Section [2.8](#) securely [16]. For the vehicles to perform the following maneuvers securely, these are assumed to be preloaded with the following secure material:

- A certificate with its identity obtained from the [CA](#);
- [PCs](#) issued by the [PCA](#);

#### 4.4.1 Platoon Creation

The Platoon creation was modified from the original [PMP](#), described in Section [2.8](#). From a security standpoint, the Create maneuver is more complex, with multiple parameters needing to be set up beforehand. At this level, it is an administrative operation performed by a new and external entity, the Administrator, in which all the Platoon parameters are defined. The [VPKIbrID](#) interactions, including the new entity, are shown in Figure [4.12](#). The Platoon parameters include the maximum Platoon size, the type of vehicles allowed, Platoon name, and which vehicles are allowed to join. Additionally to the normal parameters, at this point, the Leader needs to associate the public key and certificate that it is going to use. Thus, it can sign all the messages, including the beacons, allowing all future followers to easily recognize and verify the messages.

The administrator also needs to indicate which vehicles will be allowed to decrypt the beacons and messages encrypted using [VPKIbrID-ABE](#), by giving them permissions in the [IdM](#). Additionally, the [IdM](#) and the token can be used to attribute the different roles to the entities. In this case, the Platoon created will be called **PLATOON\_1**. The followers will have the attribute **FOLLOWER**, and the leader the attribute **LEADER**.

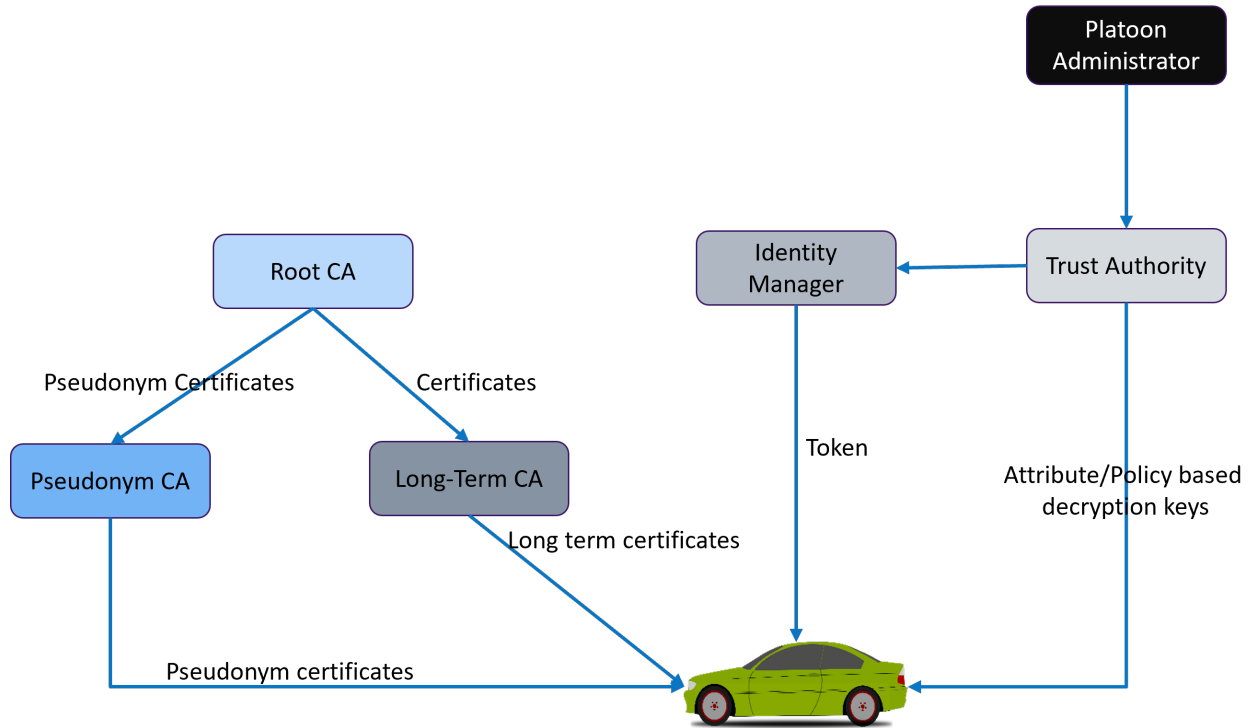


Figure 4.12: VPKIbrID Security Model Interactions with Administrator (From [16])

#### 4.4.2 Initialization

Before joining a platoon, the future follower needs to obtain some cryptographic material that will allow him to do so and be able to decipher the messages exchanged by the other elements. As the administrator already added the needed attributes for the follower to join the platoon, the vehicle can fetch the needed data.

First, the followers will obtain the ABE keys from the TA as indicated in Section 4.3.3. These will allow deciphering all the messages from the Platoon that are intended for the follower as, for example, the advertisements.

Afterward, it needs to fetch a token to be sent to the leader in order to prove its authorization to belong to that platoon. This will be requested from the IdM as indicated in Section 4.3.1.

#### 4.4.3 Disseminate Platooning Advertisements

The Platoon leader may broadcast advertisements at regular intervals so other vehicles driving on the same road can be aware of its presence. Depending on the privacy requirements of the platoon, the leader may choose to only advertise the Platoon to authorized vehicles.

To do so, the leader can use VPKIbrID-ABE [14]. The ABE mode allows the leader to send messages

ciphered to all the vehicles that share the same attributes. In this case, the leader can cipher a message using the attributes *PLATOON\_1 DRIVERS 2of2*, meaning that only vehicles possessing both attributes will be able to decipher the advertisements. The *2of2* part of the rule indicates that two attributes are needed, and both must be fulfilled to access the encrypted data. Thus, it is assured that only members of this specific Platoon (or future members) will know about its existence.

The advertisements should be signed to be verifiable by the receivers. The public key used to sign the message can be given to the follower in the initialization phase.

#### 4.4.4 Joining Platoon

A platooning-enabled vehicle that wants to join an existing Platoon needs to request permission to do so by sending a Join Request to the leader of the Platoon.

Using *VPKIbrID-ABE*, the follower can do so in such a way that only the leader of that specific Platoon will be able to decipher the request. As stated before, the Platoon has the name *PLATOON\_1*, and the leader has the attribute *LEADER*. So, the follower can use the attributes *PLATOON\_1 LEADER 2of2*, meaning that only the leader of *PLATOON\_1* will be able to read the request.

The follower needs to prove in some way that he is allowed to join the platoon. To do this, the follower includes in the request the token obtained from the *IdM* indicating that it is allowed to join the platoon. The privacy of the follower can be maintained with the *IdM* token because it does not explicitly indicate its identity. It merely needs to indicate that it has access to that Platoon. The token needs to be signed by the *IdM* to be verifiable by the leader. Although an infrastructural connection is needed for the token obtention, it is not necessary for its verification.

#### 4.4.5 Secure Message Exchange

Vehicles belonging to a Platoon can communicate in two basic ways: broadcast/multicast or unicast. They can commute between the two modes depending on the type of communication. If the goal is to communicate with all the members of the Platoon, the broadcast is advised. If the message is directed to one specific member, unicast is used.

As it is stated in [14], *VPKIbrID* provides a way to communicate in both ways. Furthermore, it concludes that *VPKIbrID-ABE* is best suited for broadcast/multicast communications, while *VPKIbrID-PKI* fits better for unicast.

So, Platoon members can use *VPKIbrID-ABE* when multicast communications are used, for example, during the dissemination of Platoon beacons and maneuvers involving more than one element (e.g., Dissolve).

Maneuvers such as leave or the parameter adjustment of a specific Platoon member can be ciphered using [VPKIbrID-PKI](#).



# Chapter 5

## Datasets Synthesis with Security Threats

The main goal of this research work is to build an IDS able to detect VANET attacks aided by ML algorithms. The latter analyzes the data provided, needing a significant amount of knowledge of network protocols and vehicle behavior patterns in order to derive a conclusion. Thus, the ML engine needs huge amounts of data, which allow them to learn normal and abnormal behavioral patterns.

However, as indicated in the SLR performed (Chapter 3), it was found that there are little to no publicly available VANET datasets. Most of the research works found in the literature use their own datasets and do not make them, nor the methodology used to obtain them, available. The availability of the dataset used to test and train the IDS is of significant importance. Without having access to the datasets, it is very hard to verify the validity of the studies published in the literature. Thus, creating publicly available datasets is essential, as they can provide a good basis for future research works.

This Chapter presents the methodology used to create multiple VANET datasets containing security threats and attacks. These synthesized datasets have been used within this work, enabling others to verify current findings. They were also made publicly available, enabling the training and validation of future research works [20].

This work tries to follow the widely known and utilized datasets Kyoto [73] and NSL-KDD [99], a set of very complete datasets that originate from the real-world traffic, but not from VANETs and, therefore, not suited for this research. The data is produced using simulation scenarios instead of real data from physical communications due to the difficulty of setting up laboratorial scenarios which are big enough to obtain all the required data. Real-world data is difficult to obtain because there is still a low volume of radio-equipped vehicles, and there is still a lack of vehicle makers willing to make them publicly available.

In the following Sections the methodology used to produced the datasets is described. Firstly, in Section 5.1, the scenarios were defined, choosing a well suited VANET application and which type of communication would be used. The selection of the geographical maps where the vehicles running the applications will be

deployed is described in Section 5.2 noticing that these directly impact the communications. For example, bigger maps with a more scarce vehicle distribution will not have problems with saturating the communication medium, but some vehicles may not receive all messages; however, smaller maps with greater vehicle density may cause too many interferences and facilitate some attacks. Section 5.3 describes the simulation, including the network and traffic simulators, the manipulation of the downloaded maps, the types of attacks implemented, and the used parameters. Finally, in Section 5.4, the data collection process and the content of the datasets are described.

## 5.1 Scenarios for Message Generation

Three different scenarios were defined: In **Scenario A**, the normal vehicles generate traffic at constant time intervals (usually referred to as CBR); In **Scenario B**, an application that generates specific application messages, such as streaming service or platooning, is used; Finally, in **Scenario C**, the vehicles disseminate generic CAMs.

**Scenario A** is one of the most common choices found in the literature (Chapter 3), probably due to its simplicity and easy implementation. In this type of scenario, an application generates a beacon at constant time intervals, usually referred to as CBR traffic [76, 80], during the simulation duration. Generally, the application does not take into account any specific vehicle parameters or the scenario, generating the messages with the same content and frequency. However, the message generation is too static so, any minor fluctuation in the generation period can be easily detectable. Consequently, attacks such as DoS are unrealistically easy to detect. The same happens with the fabrication attacks; if the message content remains the same during the simulation, any fabrication attack can be easily detected.

**Scenario B** uses proprietary applications such as platooning or a streaming service. This scenario is the opposite of the previous one, as it is too unpredictable and difficult to detect anomalies. In the case of platooning, multiple possible maneuvers can happen at random times. A very large set of data containing maneuvers and platooning beacons would be necessary to detect attacks in this type of scenario. However, this is a complex application, and implementing the application for the simulation would probably turn the data biased as the maneuvers would happen dependently on the application. In the case of the streaming service, the data is also too random. It can be composed of images, video, or sound and, its content can be of multiple formats. The bit rate can also have a huge variety.

In **Scenario C**, a more generic approach is used - the generation of CAMs. This scenario is more dynamic and realistic than **Scenario A**, as the CAM generation rate depends not only on the passage of time but also on the vehicle movement. However, it is not as random as **Scenario B** because the message generation follows a set of well-established rules, but random parameters, such as speed, heading, and

acceleration, can affect them. Moreover, the usage of CAM simulates a scenario easily implemented in the real world. It is a generic standard used by all vehicles that have communications-enabled devices. Thus, it seems to be the most suited application to use in the message collection.

## 5.2 Geographical Maps for Message Collection

The selected scenario considers vehicles generating CAMs. The CAM generation depends on the vehicle movement, which directly depends on its geographical map. So, another critical aspect for the message collection is the geographical map chosen to run the simulation.


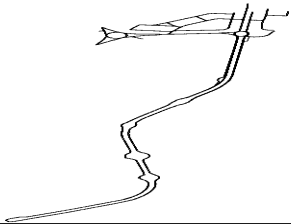
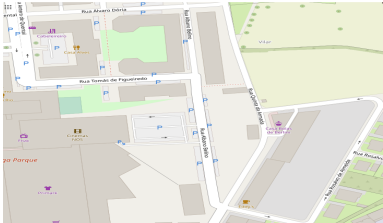
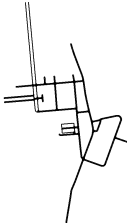
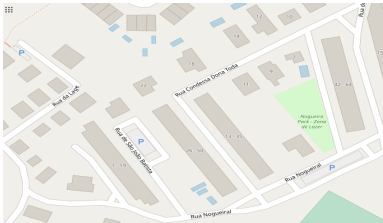
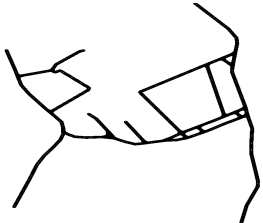
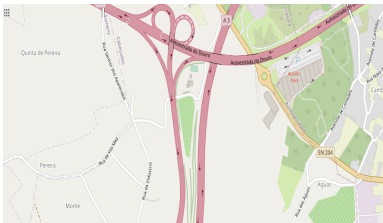


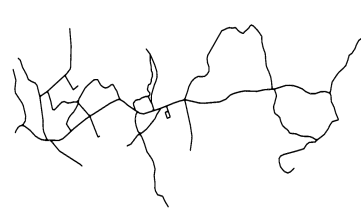
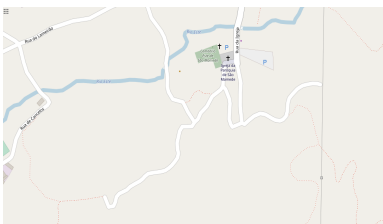

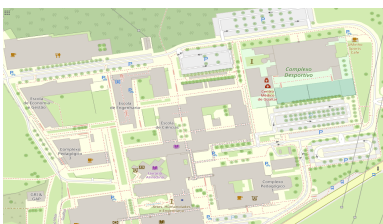
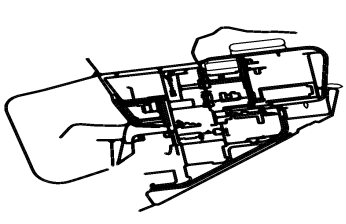
The goal was to collect multiple datasets with enough diverse data to have a robust detection. So, instead of having only one map and collecting various datasets from it, maps from 7 different geographic locations were used for data collection. Having data from maps with different traffic types, sizes, and complexity, enables the simulation of different environments and, therefore, introduces randomness to the vehicle movement and the collected messages.

The maps were collected from Open Street Maps (OSM), an online platform that provides an easy-to-use interface to download maps in OSM format from any desired location. OSM is a proprietary format but is easily translated into formats understood by most popular simulation tools.

The 7 maps were numbered from Map 1 to Map 7 (nomenclature used from now on) and are shown in Table 5.1. The maps shown in the column "Map" were the ones collected directly from OSM, the ones in the column "SUMO Map" were collected from SUMO. These can differ, depending on the number of nodes that the OSM includes of the nearby roads. Most of the collected Maps result in a bigger map than the OSM origin, with bigger roads and more secondary roads than expected. In the case of Map 4, the contrary happens. Although the selected area had a highway, it has too few nodes for it to be correctly translated into a SUMO map.

The values for the area and total road length of each of the collected OSM maps are also shown in Table 5.1. The road length of the maps varies from 2,618 to 37,248 meters, and the map area from 70,000 to 1,470,000 square meters.

Table 5.1: Road Length, Area and Maps Used for Message Collection

Map	Length (m)	Area ( $m^2$ )	Map	SUMO Map
1	11,249	140,000		
2	3,467	110,000		
3	2,618	70,000		
4	37,248	250,000		
5	7,985	440,000		
6	10,581	1,470,000		
7	15,083	130,000		

### 5.3 Simulation Setup

**VANET** simulations use traffic and a network simulator to simulate vehicle movement and communications, as the name indicates. **SUMO** and **ns-3** are two of the most popular and used simulators in this area, as shown in the performed **SLR** (Chapter 3). These can be used individually by first running the simulation in **SUMO** and then feeding the output to **ns-3** or coupled together using a third-party platform. V2X Simulation Runtime Infrastructure (**VSIMRTI**) [101] is a Java-based platform developed by Daimler Center for Automotive IT Innovations (**DCAITI**) able to seemingly couple traffic and network simulators, in this case, **SUMO** and **ns-3**.

Recently, **VSIMRTI** has been replaced by a new version called Eclipse MOSAIC [102]. It is an updated and open source version of the used software. However, it was only launched in late 2020, after most of the experimental work of this thesis had already been accomplished. So, all the experimental results presented in this thesis have been collected using the previous **VSIMRTI** version.

The map files downloaded are in the **OSM** format, which must be converted to a **SUMO** readable format. **VSIMRTI** provides scenario-convert, a useful tool that enables the generation of **SUMO** files. The indication of the vehicle route, its origin and starting time is done through a **VSIMRTI** file called mapping, which accepts those configurations, among others.

As previously mentioned, one of the goals was to have different maps with different characteristics introducing plenty of randomness into the process. Multiple parameters can influence the behavior and communications of the vehicles, such as road type, road length, total map area, and vehicle density. The collection of maps from multiple geographic locations allowed to vary these parameters. As it is possible to see, from Table 5.1, that Map 1 and 6 have more open roads fewer crosses than the others. Map 7 being an example of a urban area within a city with more complex types of traffic.

Another parameter that can influence vehicle behavior is vehicle density. Depending on this value, the communication between vehicles and movement may be affected. In this work, the density was not calculated in the traditional way, as the total sum of vehicles divided by the corresponding area (*vehicles/area*), but rather by counting how many vehicles were in the range of radio communications per second. So, when receiving **CAMs**, the receiver vehicle counts how many different sourced **CAMs** were received in a second. Even a map with fewer vehicles can have a larger vehicle density if all of them are concentrated in a particular part of the map. Table 5.2 presents the values for the average and peak density in each map. The average density varies from 5.83 to 35.05, and the peak density ranges from 14 to 95 in Maps 1 and 7, respectively. It allowed to simulate a more rural scenario in Map 1, with large stretches of road without crosses or roundabouts and most vehicles scattered, and a urban area within a city with a lot of traffic and road types, as in Map 7. Vehicle density directly depends on the radio configuration of the vehicle. The parameters used in the simulations are indicated in Table 5.3.

So, as shown in Table 5.2, the maps have different combinations of average and maximum densities,

increasing the randomness. If there are more vehicles on a particular stretch of road, the vehicles will behave differently, having lower speeds, more breaking, etc. The vehicle behavior also affects the frequency of CAM generation (Table 5.4).

Each vehicle being simulated has an application running that can disseminate CAMs. All vehicles generate these automatically by following a set of parameters, defined in the EN 302 637-2 standard [103] and are presented in Table 5.4.

Table 5.2: Vehicle Density in each Geographical Map Used for Message Collection

Map	Avg Density	Peak Density
1	5.83	14
2	18.21	28
3	14.09	19
4	12.31	31
5	7.06	16
6	7.49	21
7	35.05	95

Table 5.3: Radio Configuration of the Vehicles Used in Message Collection

Parameter	Value (dbm)
Energy detection threshold	-99
Tx gain	10
Rx gain	-10

Table 5.4: CAM Parameters Used in the Message Generation Application

Field	Value
Max Interval	> 1000 ms
Min Interval	< 100 ms
Position Change	> 4 meters
Heading Change	> 4.0 degrees
Velocity Change	> 0.5 m/s

According to the parameters in Table 5.4, a CAM is only generated if there is a position change of at least 4 meters, a change in heading bigger 4.0 degrees, or a speed change of more than 0.5 m/s. Nevertheless, the interval between CAMs must be at least 100 ms. If none of the parameters varies enough, a CAM must be generated each 1000 ms.

Each vehicle in the simulation has radio equipment and an application capable of sending and receiving CAMs. The normal vehicles will behave according to the rules presented in Table 5.4. The attackers can

perform differently depending on the attack selected: in DoS mode, they will break the rules from Table 5.4; in Fabrication attack, they will comply with the rules but send fake information.

## 5.4 Data Collection

As previously mentioned, the data collection will be driven by multiple simulations. The vehicles will be deployed into 7 different maps. All of them will have a communications-enabled device and will be generating CAMs. In each simulation, there will be normal and rogue vehicles. The normal vehicles will behave as expected, generating CAMs with true information and following the rules specified. The attackers will perform attacks at random times that will endure for random intervals. In each simulation, the attackers will all perform the same attack: DoS or Fabrication attack.

There are a great diversity of existing attacks (Table 2.1), which is continuously growing due to the discovery of new vulnerabilities or new types of strategies for the attacks. However, for this work, a subset needed to be selected. In this case, the choice has fallen on DoS and Fabrication. These are two simple attacks but are very common. So, they seem a good starting point. The application produced can then be modified to produce new attacks, enriching the datasets and update the IDS with new detection capabilities.

The datasets will be built by collecting and storing all the data exchanged by the vehicles in each simulation, originating an individual dataset per simulation. All vehicles will store both sent and received messages. The CAMs, the type of message chosen, do not have any extra field to indicate an attack. So, only the sender can know if the sent message is an attack. Thus, when storing the sent messages, the sender needs to mark those corresponding to an attack. However, only the received messages are needed for the training of the IDS, so, after each simulation, the sent messages are used to classify the received as an attack or not.

The sent messages contain the following information: **vehicle** - unique vehicle identification for each simulation with the format veh\_XX, with XX being the sequential number of the vehicle as attributed by the simulator; **time** - simulation time in nanoseconds at which the vehicle has sent the message; **isAttack** - field that indicates the attack type. The possible values are (Table 5.5): 0 (no attack); 1 (DoS), 2 (fabrication attack speed), 3 (fabrication attack acceleration) and 4 (fabrication attack fake heading);

The receiver vehicle will store more complete information, either directly retrieved from received CAMs or computed. The format of the stored information is: **senderId** - Identification of the vehicle that sent the message; **receiverId** - Identification of the vehicle that received the message; **receiverTime** - Simulation time in nanoseconds at which the vehicle has received the message; **diffTime** - Time difference between two consecutive received messages from a vehicle with the same ID; **heading** - Heading of the sender vehicle in degrees; **speed** - Speed of the sender vehicle in m/s; **longAcceleration** - Longitudinal acceleration of

the sender vehicle in  $\text{m/s}^2$ ; **generationTime** - Time at which the message was generated in nanoseconds; **elevation** - Elevation of the vehicle in meters; **latitude** - Latitude of the sender vehicle in degrees; **longitude** - Longitude of the sender vehicle in degrees; **bitLen** - Bit length of the received message; **diffPos** - Distance between the position of two consecutive received messages from the same vehicle; **diffSpeed** - Difference between the speed received in two consecutive messages from the same vehicle; **diffHeading** - Difference between the heading received in two consecutive messages from the same vehicle; **diffElevation** - Difference between the elevation received in two consecutive messages from the same vehicle; **diffAcc** - Difference between the acceleration received in two consecutive messages from the same vehicle. These parameters stored in the dataset are summarized in Table 5.6.

Most of the collected data parameters are directly taken from the vehicle simulated, such as the speed, acceleration, and heading. However, the parameters **diffTime**, **diffSpeed**, **diffHeading**, **diffElevation**, and **diffAcc**, are calculated based on each parameter from two consecutive messages received from the same vehicle. The goal was to have some variables that could produce more insight than only the value of each parameter, allowing the characterization of the vehicle's movement. For example, in a fabrication attack, the difference between the values in two consecutive messages may not be congruent. The same may happen in DoS; the value of **diffTime** will probably be less than the one from vehicles behaving normally.

The data will be collected by performing simulations in each of the 7 different maps. Each attack will be performed independently, resulting in a different dataset for each different configuration. For each attack configuration, a dataset for each map was obtained, resulting in 7 datasets for configuration and a grand total of 42 datasets. The obtained datasets were the following:

- Denial of Service (DoS)
  - 7 Datasets with periods ranging from 1 to 10% of the normal CAMs;
  - 7 Datasets with periods ranging from 10 to 20% of the normal CAMs;
  - 7 Datasets with periods ranging from 20 to 30% of the normal CAMs;
- Fabrication

Table 5.5: Codes of the Different Message Types Collected

Message Type	Value
Normal	0
DoS	1
Fabrication Speed	2
Fabrication Acceleration	3
Fabrication Heading	4



Table 5.6: Parameters of the Collected Messages

Message Type	Units
sender ID	veh_xx
receiverID	veh_xx
receiverTime	nanoseconds
diffTime	nanoseconds
heading	degrees
speed	m/s
longAcceleration	$m/s^2$
generationTime	nanoseconds
elevation	meters
latitude	degrees
longitude	degrees
bitLen	bits
diffPos	meters
diffSpeed	m/s
diffHeading	degrees
diffHelevation	meteres
diffAcc	$m/s^2$

- 7 Datasets with random speed data;
- 7 Datasets with random acceleration data;
- 7 Datasets with random heading data;

The described methodology originated multiple datasets with different types of attacks. The datasets are thoroughly analyzed in Section 8.3. Each dataset contains only one type of attack, making it simpler to test IDS implementations using different strategies, for example, detecting only one attack (using only one type of dataset) or multiple (merging multiple datasets). The datasets synthesized were submitted to a public online platform and are available at <https://zenodo.org/record/4304411> [104] (accessed 01/2022). The code used for their fabrication is available at <https://github.com/fabio-r-goncalves/dataset-collection> (accessed 01/2022). These datasets are the basis for the training and testing of the IDS designed in this work.

# Chapter 6

## Intelligent Hierarchical IDS for VANET

Recalling, the main goal of the research work was the design, implementation, validation and testing of an Intelligent Multi-layered Hierarchical Intrusion Detection System for [VANET](#)s. The [IDS](#) should detect attacks using a cluster-based structure, allowing nodes with similar characteristics and needs to be grouped.

This Chapter first describes the architecture at an abstract level, indicating the main building blocks, their roles, and cluster division. Then, in Section [6.2](#), the secure channel across the architecture is described as well as the interactions between the diverse entities needed to exchange data. Finally, in Section [6.3](#), the best-suited detection algorithms for each hierarchy level are described, as well as the roles they should perform.

### 6.1 Architecture of the Intelligent Hierarchical IDS

The architecture of the [IDS](#) is based on a hierarchy. Thus, it presents multiple levels, each composed of several clusters of entities that share the same characteristics, functionalities, and needs. Therefore, the architecture design must facilitate the detection at different levels, carefully evaluating the best functions and detection types, according to their capabilities - processing power, storage capacity, etc. - and needs - detection time, delay, accuracy - with better performance at each clustering level.

The solutions should also encompass a security framework to enable the secure exchange of information between the cluster nodes, considering the needs and requirements of [VANET](#) applications. In addition, the security frameworks should provide an underlying communication secure channel that should facilitate secure broadcast communications, authentication at multiple levels (driver authentication, entity authentication), privacy, and confidentiality, without disregarding the other requirements described in Section [2.3](#).

Figure [6.1](#) presents the architecture from the functional point of view. It includes the layer division and

the secure channel across the layers. The goal of the design was to attribute more complex and CPU-heavy functions to the upper layers, reserving quicker and lighter operations for the lower layers. Hence, taking advantage of the characteristics of the nodes at each layer and provide quick responses at the lower layers. The architecture is divided into the following levels:

- $L_0$  Each vehicle: These are the smallest cluster composed of a single entity
- $L_1$  A group of vehicles organized into a single cluster;
- $L_2$  All the vehicle clusters within a geographical region;
- $L_3$  Cluster of all geographic maps;

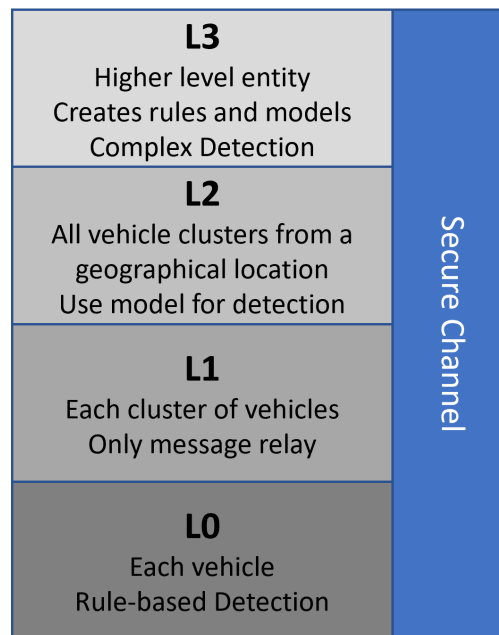


Figure 6.1: Hierarchical Intelligent IDS Security Framework Functional Architecture [23]

$L_0$  is the lowest level in the hierarchy. It is a single vehicle that is the least powerful entity in the network, containing less CPU power and storage capacity. However, these are the entities closer to all messages interchange and ITS messages dissemination. They receive the messages directly and need to analyze them as quickly as possible to allow a decision in usable time. If the detection at this level is too slow or heavy, it may facilitate a DoS attack. Hence the functionality that was chosen for this entity. The  $L_0$  entities will receive messages from the other entities and only apply Rule-based detection, one of the quickest types of detection. At this level, quick decisions may be more important than the overall accuracy. However, the Rule-based detection needs to have a very low rate of false positives when analyzing the normal messages; otherwise, it will discard authentic messages.

$L_1$  is the first level composed of more than one entity. It is formed of multiple vehicles that compose a cluster. The vehicles can be grouped, for example, according to their travel path and speeds. Facilitating the sharing of information during bigger intervals of time, as they will be traveling in the same route in communication range from each other. This level will not perform any special type of security threat detection. The entities that compose it are the same type as the node before and, thus, do not present any advantage of CPU power or storage capacity. The only difference is the time the detection takes to reach the entity that needs the response. So, at this level, all the entities will send the messages to the cluster head (for instance, the platooning leader), and this entity will relay the messages to the next level in the hierarchy.

$L_2$  is a cluster with entirely different characteristics from the previous clusters. It is composed of infrastructural nodes capable of much more complex and heavy operations. Additionally, it can have hardwired communications with the above level, being able to communicate much more data. The RSUs will receive the messages from the nodes lower in the hierarchy and simultaneously analyze and forwarding them to the next hierarchy node. If an attack is found, it is communicated immediately to the sender. They can also trigger a system-wide warning and warn the upper-level nodes so that the IDS may black-list the attacker. These nodes will use more complex and powerful ML algorithms to detect the attacks and trade detection time with better accuracy. The RSUs are not as close to the attacks and do not need to identify and detect any attack immediately; it is rather more important to be able to accomplish better accuracy at this level.

$L_3$  level is the highest level in the hierarchy. It comprises the most powerful entities that may carry much more complex operations and store messages, models, and rules. In addition, these entities are generally powerful backend servers with high-performance CPUs. Hence,  $L_3$  entities can collect the messages sent from all the lower-level nodes and analyze them, creating ML models and rules to be used by the other levels. These high-level entities are very far from the nodes needing detection; thus, the detection time is not an issue here. They can also perform "offline" detection, using the result to trigger a system-wide response, black-list attackers, and, if needed, notify the authorities. So, at this level, the goal is to use the more complex detection; this detection takes time, it is certainly slower but may detect attacks undetectable by other entities.

The resulting architecture is presented in Figure 6.2. The Figure is divided into two blocks, representing the network level at which each node is located. The levels  $L_0$  and  $L_1$  are in the VANET. The node  $L_2$  connects the two types of network, and the node  $L_3$  is located on the Internet, with only cabled communications; the right side depicts the communication of the CAM messages from the sending vehicle to node  $L_3$ . Each time the message has a yellow padlock, it means that it is encrypted.

As shown in the Figure, the multiple nodes also forward the messages and group them into bigger blocks; the left side depicts the communication of the models and rules from the  $L_3$  nodes to the  $L_0$ . The rules are shown in orange and the models in green. Additionally, Figure 6.2 shows which function each cluster has on top of forwarding messages. Nodes in  $L_0$  use the rules to analyze the collected data and, in  $L_2$ , they use the ML models.

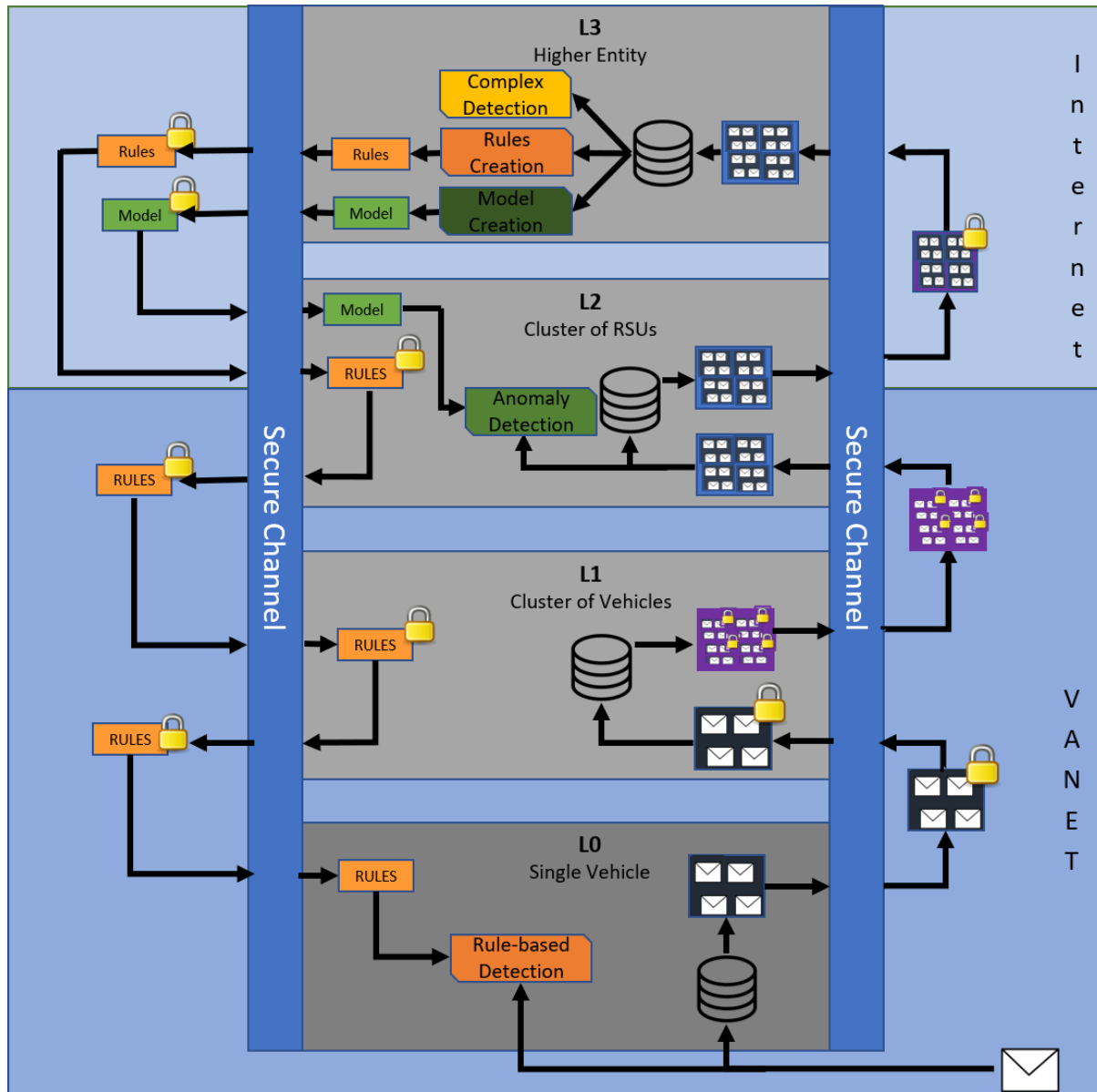


Figure 6.2: Hierarchical Intelligent IDS Architecture

## 6.2 Interactions and Secure Message Exchange

The proposed IDS uses a hierarchical architecture to provide cooperative detection, as all nodes cooperate to detect the attacks. However, despite IDSs being able to detect attacks, they do not provide tools for secure communication, which are crucial for the correct function of the overall architecture. Security is one of the most crucial components in a system involving communications. Without it, the several architecture entities would not be able to verify the veracity of the received messages and consequently may use fabricated messages to train the IDS. Furthermore, the lower layers would not be able to verify the veracity of the received modified models incapacitating them from correctly detecting attacks.

The security implementation in this architecture is represented in Figure 6.2 by the blue rectangle across all the layers. It provides multiple communication types allowing entities to take advantage of the one best suited for the situation.

Let us recall that VPKIbrID entails two different modes: one the of the VPKIbrID modes uses PKI, more suited for unicast communications, with only one receiver for the sent message. The other uses ABE, being useful in situations where there are several targets for secure message transfer. Using VPKIbrID-ABE, the sending entity may encrypt the message for multiple entities, using their corresponding attributes. In this case, the VPKIbrID-ABE seems to be the better suited, as most of the communications will have multiple targets, at least when happening between clusters. However, this mode is slower and heavier than the PKI mode. Still, the VPKIbrID provides the possibility to use key caching, significantly increasing its performance. For example, when receiving CAMs, the vehicles can encrypt the received messages using the attributes " $L_2 L_3 1of2$ "; thus, all the clusters  $L_2$  or  $L_3$  entities can read the messages. Field 1of2 means that only one attribute needs to be fulfilled. It is even possible to use attributes like " $L_2-company1 L_3-company1 1of2$ ," specifying that only the nodes from clusters  $L_2$  or  $L_3$  from any corporation (or institution) identified as "*company1*" can decrypt the message.

The multiple entities in the architecture need to communicate to exchange the information necessary for the detection. The lower entities send the received messages for the upper nodes (Upstream communication, Section 6.2.1). Finally, the upper-level entities send the models or rules created for the lower entities (Downstream communication, Section 6.2.2).

### 6.2.1 Upstream Communication

The upstream communication refers to messages sent from the nodes in level  $L_0$  to the upper nodes in level  $L_3$ . In this case, the vehicles receive the CAMs sent from other vehicles and, after analyzing them, send them to the upper levels. The messages are going to be processed both in nodes  $L_2$  and  $L_3$ . So, the encryption mechanism needs to be able to encrypt the message so that entities in both nodes can decrypt

it without needing multiple encryptions. Otherwise, the vehicles, an already low resource entity that needs to be constantly analyzing the received messages, would also be in need to encrypt the messages multiple times, would easily be overwhelmed. So, [VPKIbrID-ABE](#) seems to be a very good choice for this scenario, mainly when taking advantage of the key caching mechanism. The entities exchanging messages have the same attributes during long periods, being the perfect scenario for key caching. Both  $L_2$  and  $L_3$  nodes can detect attacks in the messages received and they can send the warnings encrypted using [ABE](#). Thus, allowing nodes on  $L_0$  and  $L_1$  to know about the attack.

Figure 6.3 shows the complete interaction from the vehicles ( $L_0$ ) to the high-level entity ( $L_3$ ) whose description follows, identifying each interaction by the corresponding number in the Figure. Firstly, the  $L_0$  vehicle receives a [CAM](#) (1), and using the rules sent by the  $L_3$  entity, verifies if it is any attack (2). If so, the message will not be processed by the vehicle (3); the message then is ciphered (4) and sent to the upper level (5). The level  $L_1$  only acts as a relay. So it will do so for the level  $L_2$  (6).  $L_2$  will decipher the message (7) and verify if it is an attack using the models built by the level  $L_3$  (8). If an attack is detected, the other entities are warned about the attack. So, a message is sent ciphered using the attributes " $L_0 L_1 1of2$ " (9), enabling all the lower entities to read the message. Regardless of being an attack, the message is relayed to the upper level (10). The next level will then decipher the message (11), verify if it is an attack (12), and add it to the dataset (13). If an attack is detected, a system-wide warning is triggered (14).

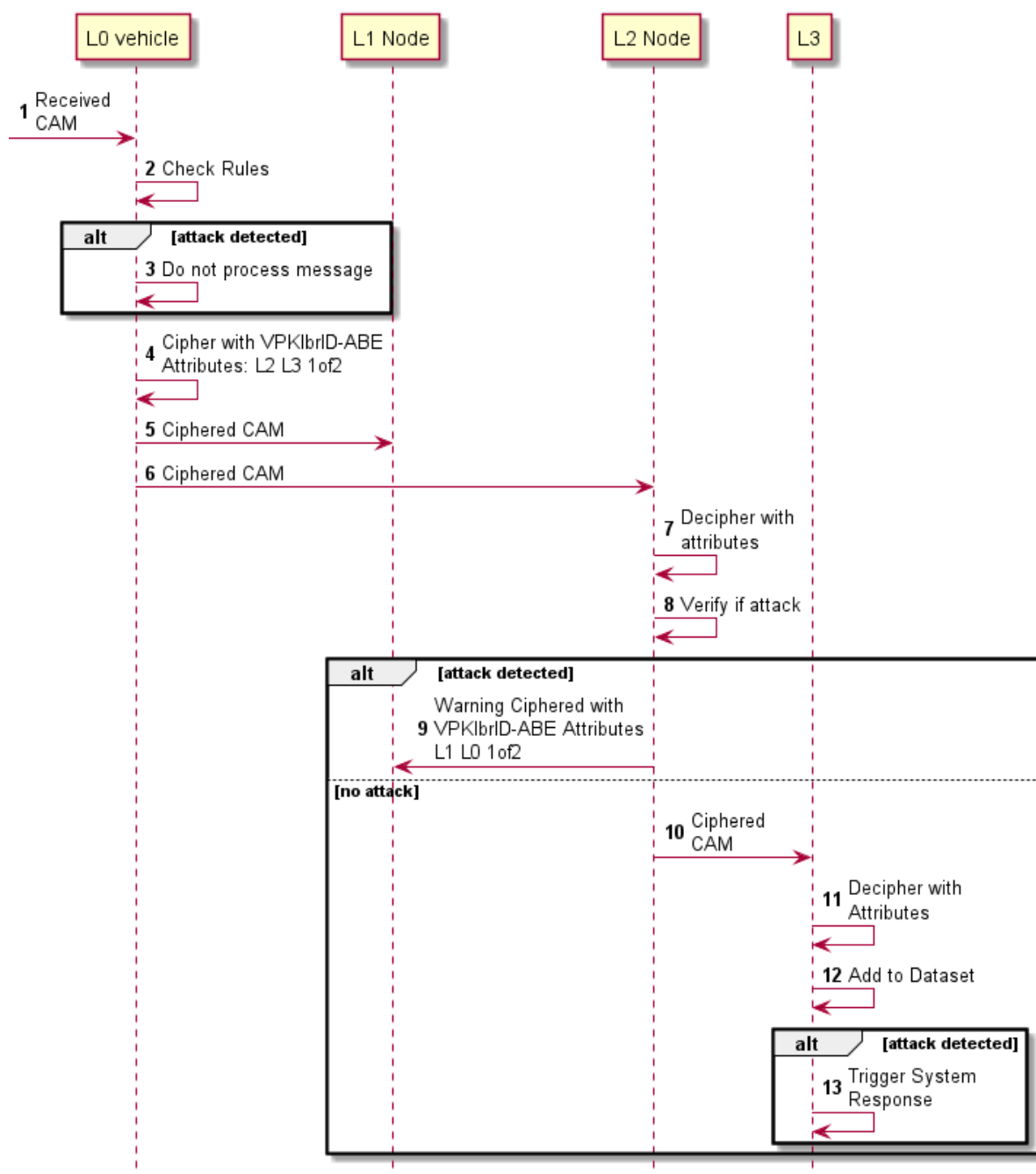


Figure 6.3: Upstream Communication and Processing of the Received CAMs



### 6.2.2 Downstream Communication

The downstream communication is exactly the opposite of the previous type. This communication process happens when the higher-level nodes,  $L_3$ , need to communicate the computed rules and ML models to the lower-level nodes, in this case, to the  $L_0$  and  $L_2$  nodes, respectively. It is less complex than the upstream communication because there is no warning going in the opposite direction. This communication mode also takes advantage of the encryption to multiple targets provided by the ABE mode of the VPKIbrID security framework. Downstream communications enables the higher-level entities to send the models and rules for all the nodes in  $L_2$  and  $L_0$  nodes simultaneously using the attributes that describe the cluster where they are located instead of the node itself. Otherwise, the  $L_3$  nodes would have to send messages individually to each target, with the risk of saturating the communication channels.

The top-level entities can then use the attributes " $L_2$  1of1" and " $L_0$  1of1" to cipher the created model and rule, respectively. This rule ensures that only the nodes in level  $L_2$  will receive this particular model, and the vehicles in  $L_0$  will receive these rules (1of1 indicates that only one attribute is needed, but it must be fulfilled to access the encrypted data). The process is shown in more detail in Figure 6.4. Before the process is started, node  $L_3$  needs to receive multiple CAMs from the lower-level nodes (1). These will be used to build a dataset (2), which will be analyzed to create rules and models (3) for the other levels. The models will be ciphered with different rules for the different levels (4 and 5). After correctly secured, it will be sent for level  $L_2$  (6). This level will decipher its model (7) and relay the rules for the lower level (8). The level  $L_1$  does not perform any detection and thus does not need any rules or models, and forwards (9) the received rules for the level  $L_0$ . These nodes will be able to decipher the rules (L0).

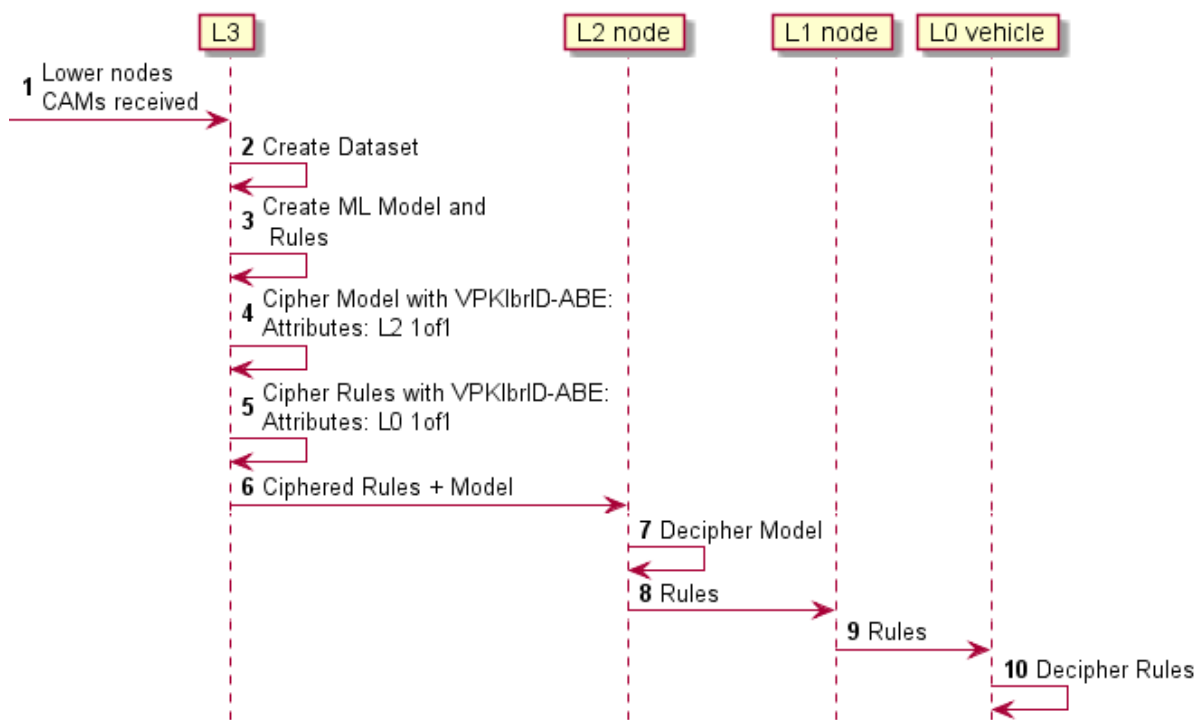


Figure 6.4: Downstream Communication of the Produced Rules and Models

## 6.3 Detection Algorithms and Roles

The goal of dividing the architecture into multiple cluster levels was to attribute different roles and capabilities to each entity, depending on its capabilities and needs, resulting in four different levels. The capabilities and needs grow inversely from each other, as shown in Figure 6.5, higher levels have more capabilities and fewer needs in terms of detection time. These entities are far from the entities receiving the CAMs and are not able to respond timely (near real-time responses). The lower entities need to have a quick decision but have fewer capabilities.

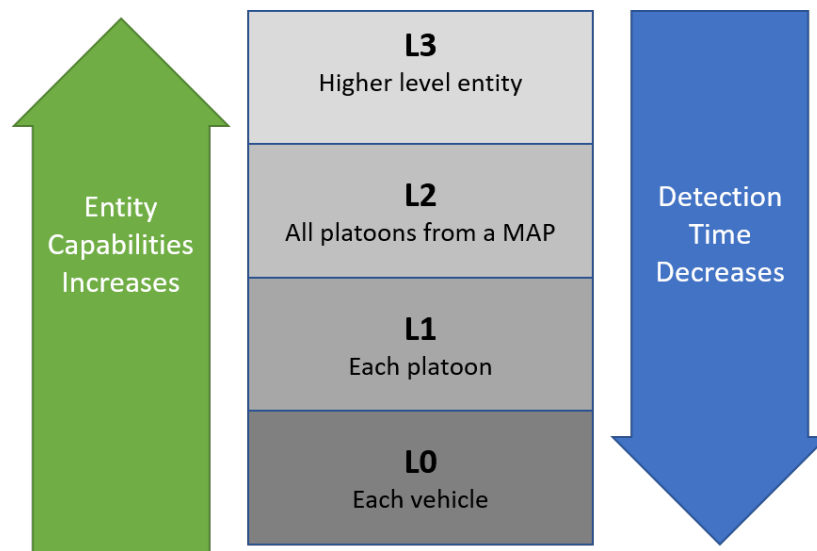


Figure 6.5: Cluster Level Needs vs. Characteristics

### 6.3.1 L0 Detection

The first level, with fewer capabilities but with higher speed demand, is the  $L_0$ . These nodes are simply vehicles on the road. Although new advancements in technology may provide better communication devices with more computing power, these nodes are usually less powerful than those in the above nodes. And, additionally, these receive many messages from the surrounding vehicles; being an easy target for DoS, any more computation power or time in the message analysis may be enough to overwhelm them. So, this entity should also not have the weight of building rules or ML models, and, at this level, the detection tool should be as quick and light as possible.

The results presented in Section 8.4, indicating the performance of each algorithm, show RF with the best overall performance. However, this algorithm is heavy, slow, and needs some computing power. So, a lighter algorithm should be equated at this level. One of the algorithms also tested was Decision Stump (DS).

It is an algorithm that generates rules that other nodes can easily use for detection. It is a one-level decision tree, and it creates basic rules with only an "if" statement.

DS presents a good tradeoff between speed and accuracy. On the plus side, it detects DoS with an accuracy of 0.94 and Fabrication with the tampered speed with 0.47 accuracy. This algorithm also classifies almost all the normal messages with a very low FPR, not discarding authentic messages. It is a very light and quick algorithm, and even without having the best accuracy, it seems a good choice to be used as a first defense line. The nodes at level  $L_0$  will also send the received CAMs to the nodes above for further analysis.

### 6.3.2 L1 Detection

Level  $L_1$  is the next level on the hierarchy. It is a group of vehicles organized into a cluster. Hence, this has very similar characteristics to the lower cluster as all its entities are still vehicles.

The clustering algorithm used is out of the scope of this work. However, the algorithm should try to group vehicles that travel the same path with similar speeds, enabling the vehicles to be in communication range as long as possible. Additionally, and perhaps more importantly, the clustering algorithms should consider the communication capabilities of the vehicles and, if possible, group vehicles with fewer communication capabilities with others with better equipment. Thus, the better-equipped vehicles may be transformed into cluster-head to gather all the information and forward it to the upper nodes. Ge et al. [105], Shahwani et al. [106] and Cheng et al. [107] are examples of suitable algorithms for the creation of clusters in VANETs based on the location and mobility of the vehicles.

This level does not present more computing power or storage capacity than for the level  $L_0$ . Thus, using the same algorithm in two consequent levels does not seem useful, as the rules would probably detect the exact same messages. So, the nodes on this level will only perform forwarding of the received messages to the nodes above. Thus, the best seems to be; all nodes communicate the received messages to the cluster-head (the platooning leader), then group them together and send them to a node in level  $L_2$ .

### 6.3.3 L2 Detection

Level  $L_2$  is the first cluster with infrastructural entities. These entities may have much more CPU power and storage capacity as they have few power limitations. However, these are still not the most powerful entities. The RSUs are mainly communication entities and not exactly made for processing information.

The decision at this level does not need to be as fast as in the level  $L_0$ . Although these entities are close enough that they may be able to produce decisions in useful time, this will not be immediate because of the time needed for the communications. So,  $L_2$  nodes can use more heavy ML algorithms focusing on

detection accuracy instead of speed. Nevertheless, the  $L_2$  nodes will still forward all the received messages to the above nodes for a deeper analysis and the creation of rules and models. Therefore, even with more power available, these nodes are still not the best choice for creating models and rules. The main issue at this level is the narrowed vision of the network.  $L_2$  nodes do not have as wide a view of the network as  $L_3$  and may create biased models that would tamper with the detection in the lower-levels, creating models with low accuracy, as shown in Section 8.3.2.

This level may use a more complex algorithm that is more CPU-heavy. In Section 8.4 are the results obtained with the multiple algorithms used. The one presenting the best accuracy is RF, with the best overall performance with low false positives. It has an accuracy of 0.96 with an average TPR of 0.85 and only 0.01 in FPR. This algorithm is particularly good at detecting DoS and fabrication attacks with the tampered heading. In addition, the very low FPR is very good because it indicates that almost no normal message is wrongly classified as an attack, decreasing the possibility of discarding normal messages.

### 6.3.4 L3 Detection

Level  $L_3$  is the highest level in the architecture and, thus, it has the most powerful entities with the most storage capacity. Furthermore, the  $L_3$  entities are all infrastructural, with all the communications made through high-capacity cabled connections. Hence, not having any problem with the size of data to be transmitted. Moreover, these are backend servers designed for complex and heavy computations. So, these can carry multiple operations.

First, these receive and analyze the data sent from the levels lower in the hierarchy, storing it to analyze further or prove detected attacks. The data storage also allows "offline" detection using more complex ML algorithms. However, due to the sheer size of the data received at this level, the detection will also be slower.

Therefore, the  $L_3$  level has the perfect conditions to use ensemble detection. It is a more complex type of ML that uses multiple algorithms to detect attacks. This case uses Multilayer Perceptron (MLP), RF, and J48, combined using the custom stacking algorithm. It has a small increase in performance, with a higher TPR and smaller FPR than the RF algorithm used in the lower level.

At this level, it does not make much sense to try and warn the vehicle that received the message because it may not be possible to do so in a usable time. However, it can trigger a system-wide response, blacklisting the attacking vehicle and, if needed, warn the authorities.

However, and perhaps the main job  $L_3$  level is to create rules and models to be used by the lower-level nodes. As previously mentioned, the entities forming the  $L_3$  level are the more powerful in the architecture with a wider view of the overall system. Thus, they can detect attacks much more accurately, as shown in Section 8.3.2, without compromising their performance. So, due to their more complete vision of the system, they are

more suited to analyze the data and create models and rules as this is a more CPU expensive operation than using the models or rules for the detection. These models can be constantly updated and sent to the nodes lower in the hierarchy.

## Chapter 7

# Implementation of VANET applications and security mechanisms

The proposed Hierarchical IDS Architecture comprises multiple parts, the security model, the vehicles, the application that generates CAMs and attacks, and the ML algorithms. The tools and overall architectures were assessed by implementing the needed components and analyzing the produced results. This Chapter describes the implementation of each component and tools utilized in this work.

First, a use-case implementation of the Hierarchical Intelligent IDS architecture is presented using the tools and algorithms analyzed in this thesis work and applying them to the abstract architecture designed in Chapter 6. In Section 7.2 the implementation of the VPKIbrID security model is described. It provides a secure channel that allows all the entities in the architecture to securely exchange the needed data, protecting the transmission of ML rules or models, or messages received. Then, Section 7.3 describes the application of the security model to a specific use case, through the implementation of a secure platooning . It also includes the implementation platooning itself. In Section 7.4 is described the implementation of the agnostic middleware used to provide easy and seamless communications between third-party applications and OBUs, enabling testing with multiple OBU brands. The collection of the multiple datasets was made through an application for VANET able to produce CAMs and attacks to enrich the dataset and is described in Section 7.5. In Section 7.6, it is described the implementation of the multiple ML algorithms used in this research work. Finally, Section 7.7 describes a real-world platooning implementation using real physical communication devices and vehicles in a controlled test environment.

## 7.1 Hierarchical Intelligent IDS Architecture an Application Use-Case

In this Section, the components described and chosen in Chapter 6 are organized into a more refined architecture. This includes the framework for secure communications across all the levels of the architecture, the role for each entity, and the ML applied at each level. The goal was to design an easy to deploy architecture that took advantage of the multiple layers to attribute roles well suited for the characteristics of each level's entities. Furthermore, the ML techniques that could balance accuracy with each entity's capabilities are carefully chosen, providing good accuracy without overwhelming the nodes at each level. Finally, the security model was chosen based on the security features provided including, authentications of drivers and vehicles and confidentiality and privacy. Also, the choice was impacted by the communication modes offered by the security framework chosen, mainly the encryption capability for multiple targets. The architecture with all the components, roles, and technologies is shown in Figure 7.1. It shows a real-world use case implementation, using the platooning as the most basic cluster and, at the level  $L_2$ , a cluster between multiple RSUs. The latter may be multiple RSUs that constitute a specific geographical map working together to detect attacks and gather messages. The higher entity is represented by the cloud in the infrastructural network.

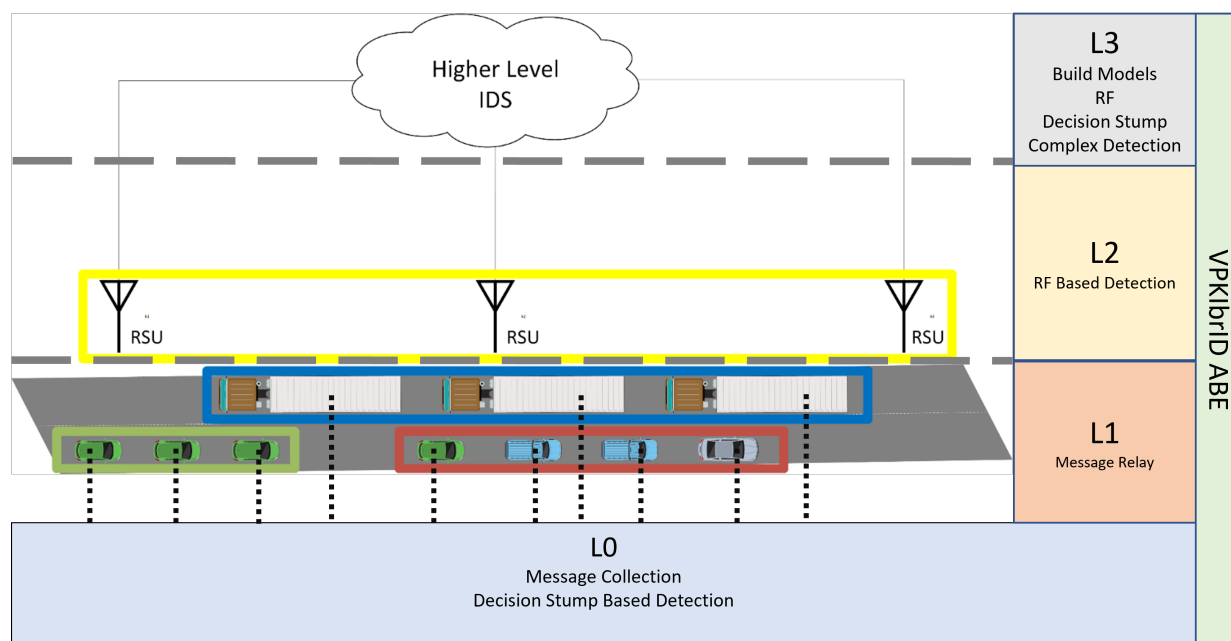


Figure 7.1: Intelligent Hierarchical IDS Architecture - Use Case

The level  $L_0$  entities are the multiple vehicles on the road; these can be road vehicles, from truck to passenger vehicle or motorcycle. Their functionality and detection type are indicated in the bottom blue rectangle. These will perform message collection, collecting the CAMs sent by other vehicles. The  $L_0$ 's nodes will be the first line of defense, using DS, the lightest and quick detection type.



Then, the green, red and blue rectangles represent multiple  $L_0$  clusters. These are Platoons of vehicles that can be composed of different types of vehicles. For example, blue Platoon is formed exclusively of trucks, while the other two can encompass multiple vehicle types. These will only act as message relays (orange square on the right), forwarding information from lower to upper nodes and vice-versa.

The yellow rectangle cluster represents the  $L_2$ , grouping multiple RSUs together. These will receive messages from the lower-level nodes and use the RF algorithm to perform a more robust detection. These are the first infrastructural entities in the architecture.

Finally, the higher level,  $L_3$ , is represented by a cloud as this is usually composed of high-capacity backend servers similar to a cloud. It is the level with the most powerful entities being able of more complex operations. So, the entities will perform the most complex decision using an ensemble learning algorithm at this level. Furthermore, they are responsible for generating models and rules for the lower-level nodes, taking advantage of their more powerful CPUs.

The green rectangle, vertical on the right, across all levels represents the security framework. Although the security model that was chosen, VPKIbrID, has more than one encryption mode, the VPKIbrID-ABE seems more indicated to communicate between the layers as most of the messages sent over the network have multiple targets. The model should be used with the key caching, which enables much faster and lighter encryptions. However, the nodes can use the PKI mode to communicate between themselves, as described in Section 4.4.

## 7.2 Implementation of VPKIbrID Security Model

VPKIbrID is the security model that serves as the basis for secure communications across multiple entities in the architecture. It provides strong authentication while also providing privacy and confidentiality. It is composed of multiple components used to provide different cryptographic materials needed to secure the communications. Each entity was implemented independently and can run standalone.

The VPKIbrID-ABE and VPKIbrID-PKI were implemented as a Java library, easily imported into any application. Java [108] is a platform-independent language that runs in most client-side devices. Moreover, it is easily converted into android for mobile applications.

All the VPKIbrID components also use Maven [109] as a project management tool that manages the used libraries in the applications, simplifying the development and distribution. The following Sections describe the implementation of the multiple VPKIbrID entities.

### 7.2.1 Certificate and Pseudonym Certificate Authority

The purpose of the **CA** and **PCA** is ultimately the same, generate certificates for the entities; the only difference is the parameters included in the certificates generated. The **CA** generates an **LTC** that represents the true identity of an entity by linking its private and public keys to the identification. The **PCA** generates **PC**; these are linked to the **LTC** only by the **PCA** and use pseudo-identifications instead of the real identification. These provide means for entities to authenticate themselves, protecting their privacy. They may possess multiple **PCs** and use them interchangeably to prevent tracing across the network. Also, the **CA** uses a self-signed certificate when authenticating itself to others, while the **PCA** uses a certificate signed by the **CA**. Thus, their implementation is very similar and will be described in conjunction. These are command-line applications that read a set of parameters from a text file used for certificate generation. An example of the file is shown in Figure 7.2.

```

CN=CAR2X-CA
OU=CAR2X
O=Universidade do Minho
L=Braga
S=Gualtar
C=PT

VALIDITY_CA=12
VALIDITY=12
PUB_KEY_ALG=RSA
SIG_ALG=SHA256withRSA
KEY_SIZE=2048

CA_JKS=ca/ca.jks

CERTS_FILE=cacerts.txt

```

Figure 7.2: **VPKibrID CA** Example Configuration File

In this case, the parameters of the certificate indicate: the Common Name (**CN**) of the owner is CAR2X-CA, its Organizational Unit (**OU**) is the CAR2X, which belongs to the Universidade do Minho Organization, Located in Braga in the Gualtar (**S**). The country (**C**) is Portugal. It has a validity of 12 months and uses an RSA key with 2048 bits. The certificate is signed using SHA 256 and an RSA key. The last two parameters indicate where the CA certificate is located, which will be used to sign the certificates, and the **CERTS\_FILE** is a file containing a list of multiple **CNs** used to generate the certificates.

The application will generate a set of certificates and their correspondent private keys. In the **CA** case, the **LTCs** will be sent to the requesting entity through a secure offline connection, preferentially using different

methods for the certificate and the private key. For the PCs, the communication method should be the one described in Section 4.3. All the certificates generated from both PCA and CA, are stored in a Java Key Store (JKS) secured with a password.

## 7.2.2 Identity Manager

The IdM is the entity responsible for the token generation. The token provides access control and authorization while providing privacy. So, any entity can prove its right to access a resource without needing to provide its identification. However, the resource owner can verify the authenticity of the token by verifying the included signature.

The IdM implementation used JWT [35] for the token generation. These use JavaScript Object Notation (JSON)[110] to describe the token's information and provide mechanisms for the signature, providing easy verification across the network. Moreover, the JWTs are widely used on the internet, being the basis of OpenID connect. Thus, there are multiple implementations and libraries available for a multitude of programming languages, making its implementation easier.

The IdM application is a standalone Java application that generates JWT tokens based on the attributes read from a file. This application was made for testing only, and thus, the functionality is simplified; in a real-world application, the IdM should contain a database with all the system entities. Then an administrator, or any entity with permission to attribute roles or attributes to other entities, would do so. However, the IdM has all the operations needed to generate and load all the tokens and cryptographic material.

So, firstly the IdM reads a file containing the configurations of the IdM, shown in Figure 7.3. It contains the entity that is issuing the token (**TOKEN\_ISSUER**), the expiration time of the token (**TOKEN\_EXP\_MIN**), and the time in minutes needed for the token to be valid after its generation (**NOT\_VALID\_BEFORE\_MINUTES**). Additionally, it contains the JKS name where the IdM certificate (**IDM\_JKS**) is stored and the alias used to store the key in the JKS. The CA signed the IdM certificate for it to be verifiable by the other entities in the architecture.

```
TOKEN_ISSUER=CAR2XIDM
TOKEN_EXP_MIN=43200
NOT_VALID_BEFORE_MINUTES=1
IDM_JKS=IdM . j k s
IDM_ALIAS=IdM
```

Figure 7.3: VPKIbrID IdM Example Configuration File

Then, the IdM reads the attributes corresponding to entities requesting the token and inserts them into the token. The token is generated for a specific goal, which is also included in the token. So, when trying to

access a resource, the resource can verify if the token was originated with that end.

Figure 7.4 presents one token generated by the [IdM](#). In this case, it was a token generated for the obtention of the [TA](#) decryption keys, indicating that the owner of the token has the attributes `platoon_a` and `driver`. The fields of the token represent: what is the token intended for (**aud**), who required the token (**sub**), the attributes of the requiring entity (**attributes**), the identity of the [IdM](#) that generated the token (**iss**), the expiration date of the token (**exp**), the data at which it has been issued (**iat**), the data from which it can be used (**nbf**) and finally a unique identifier of the token (**jti**). The token also has a header, which indicates the algorithm used for the signature, a unique identifier for the key used. The token, including the header and the signature, is encoded in Base64 to be safely transmitted over any communication channel.

```
{
  "aud": "TA",
  "sub": "null",
  "attribute": [
    "platoon_a",
    "driver"
  ],
  "iss": "CAR2XIDM",
  "exp": 1499644744,
  "iat": 1497052744,
  "nbf": 1497052684,
  "jti": "CJlywW1Lncn4fNlkXqm_Hg"
}
```

Figure 7.4: [VPKIbrID IdM](#) Token Example

### 7.2.3 Trusted Authority

The [TA](#) is one of the critical components for [ABE](#) encryption. It has two main goals, generating the public parameters and the decryption keys. The public parameters are used by any entity encrypting data. These are used in conjunction with the data's access control attributes to generate an encryption key. The [TA](#) is the only entity that can generate the decryption keys. It uses the attributes sent by the entity wanting to decrypt the data and generates a key that enables it to decrypt the data.

The [TA](#) was also implemented as a command-line application. Instead of receiving the tokens through the network, it simulates them by reading them from a file. So, the [TA](#) first starts by loading the secret keys needed to generate the decryption keys and then loads the [JWT](#) token. Using the [IdM](#) certificate, it can verify the token validity and extract the attributes from the token. Finally, it outputs the decryption keys to a file. The application should receive and send the requests over the network in a real application, secured using

VPKIbrID.

ABE is a less known and used cryptographic scheme. Thus there are no widely used libraries being usually developed by single developers. In this implementation, the library is provided in <https://github.com/junwei-wang/cpabe>. It is a realization of the Cipher-Policy Attribute-Based Encryption (CP-ABE). The library is based on the Java Pairing-Based Cryptography (JPBC) library needed for the mathematical and pairing operations involved in the ABE cryptography.

### 7.2.4 VPKIbrID Message Encryption

VPKIbrID allows entities to exchange data with each other over an insecure network securely. It provides two encryption mechanisms with different characteristics, enabling different strategies that better suit the environment.

VPKIbrID is implemented as a Java library to provide a seamless way to use in any application. It reads the configurations from a configuration file (Shown in Figure 7.5), indicating the location of the client certificate (CA\_JKS) and the CN used to store the certificate. Moreover, the application using this configuration file will have the following configurations: **SHA-256** as the hashing algorithm for the message to be signed (**CLIENT\_HASHING\_ALG**), the public key algorithm for the encryption will be RSA (**CLIENT\_PUB\_ENCRYPT\_ALG**), an **AES** symmetric key (**SYMMETRIC\_KEY\_ALG**) with 128 bits (**ABE\_SYMMETRIC\_KEY\_SIZE**) and using the **AES/ECB/PKCS5Padding** as the algorithm to encrypt the data in each message (**SYMMETRIC\_ENC\_ALG**), and using **HmacSHA1** for the symmetric key algorithm (**CLIENT\_SYMM\_SIGNATURE\_ALG**). In this specific configuration, the use of key caching is disabled (**CACHE\_ABE\_KEYS**).

```
CA_JKS=client/veh_0.jks
CN=veh_0
CLIENT_HASHING_ALG=SHA-256
CLIENT_PUB_ENCRYPT_ALG=RSA
ABE_SYMMETRIC_KEY_SIZE=128
SYMMETRIC_ENC_ALG=AES/ECB/PKCS5Padding
SYMMETRIC_KEY_ALG=AES
CLIENT_SYMM_SIGNATURE_ALG=HmacSHA1
CACHE_ABE_KEYS=false
```

Figure 7.5: VPKIbrID Library Example Configuration File

The created library provides a set of tools to encrypt data. First, the data to be sent is signed using the parameters read from the configuration file. The resulting signature is used in conjunction with the data, the sender certificate (PC or LTC depending on the scenario), the receiver public key, and the signature and signature algorithm to build the message that will be encrypted. In the case of ABE, instead of the receiver

public key, the Attributes are used. This message will be one of the parameters of the method created to encrypt the data. The encryption method will also receive a symmetric key and the encryption algorithm to perform the encryption. The result will be a [VPKIbrID-EM](#) message specified in Section 4.3.

The decryption process is simpler, it only needs the certificate of the private key of the receiver, in the case of the [PKI](#). In the case of the [ABE](#), it needs the public parameters and decryption key obtained from the [TA](#).

## 7.3 VPKIbrID Secured Platooning Implementation

Platooning is a complex application that needs a constant exchange of information with other vehicles belonging to the same platoon. Thus, to function properly, it needs a secure way for its entities to communicate. These need strong authentication, confidentiality, and privacy. So, [VPKIbrID](#), the proposed secure model, was applied in platooning to test its functionalities. In this Section first is described the implementation of the platooning application and, then, the implementation of the [VPKIbrID](#) model in the platooning, assuring the communications security.

### 7.3.1 Platooning

Platooning is an [ITS](#) application that allows vehicles to travel in a convoy manner with constant gaps and speeds. The application has two types of entities: the Leader, who controls all the Platoon maneuvers, and the Followers, who comply with the orders of the Leader. Next is described the implementation of the platooning application in simulation (A real-world implementation was also made and it is described in Section 7.7).

Platooning is a communication dependent application as its members need to communicate with each other to coordinate themselves. So, for its implementation in simulation, a traffic simulator and a network simulator are needed. For this implementation, [ns-3](#) and [SUMO](#) were chosen; these are two of the most popular simulators available that are free to use and have a strong community updating them constantly.

However, using a framework able to couple both of them together can simplify and offer new functionalities to the implementation. [VSimRTI](#) was chosen the framework chosen. It is a Java-based framework that separates the development of applications from the configurations and network. It provides an easily accessible and well-documented interface for communications and a simple way to access most of the vehicle parameters available and to [SUMO](#)'s Traffic Control Interface ([TRACI](#)).

[VSimRTI](#) has multiple configuration files that define multiple parameters, from the radio and network to the characteristics of the vehicle. One of these files is the "mapping" file. It allows the definition of multiple

parameters, including the type of the vehicle, the application that it will run, and the path it will take (based on the SUMO files). One of the parameters indicated in this file is if the vehicle will belong to a platoon or not and at which platoon (if multiple are defined).

The application is the same for the Follower and Leader, as any vehicle can become both, depending on the configurations. Each vehicle starts in a neutral state, not running any specific application or protocol. Then, it uses the parameter indicated in the mapping file to choose which configuration file it will read. This parameter can be, for example, **NON\_PLATOON**, **PLATOON\_VEHICLE\_1**, etc. If the **NON\_PLATOON** is attributed, it means that the vehicle would not be part of any platooning. The content of the configuration file for the platooning vehicles is shown in Figure 7.6.

```
is_platoon=1

group_name=GROUP_2
leader=veh_9

leaving_v=veh_10 , veh_11
left_color=YELLOW
leaving_time=230,450

dissolve=550
```

Figure 7.6: Platooning Example Configuration File

The most significant configuration file parameters indicate the following: the vehicle will belong to a platooning (**is\_platoon**), called GROUP\_2 (**group\_name**) with the Leader veh\_9 (**Leader**), the vehicles veh\_10 and veh\_11 will leave the platooning at the instants 230, and 450 and the platooning will dissolve at the instant 550.

The implemented application works by going through different states depending on the maneuver that is happening. The multiple states are shown in Figure 7.7.

The first decision is made after the initialization; the application may then become a **Platooning** vehicle or a **Non-Platoon**. If it becomes a **Non-Platoon**, it will not interact with the platooning vehicles, running its pre-determined path with the configurations given in the mapping file.

If the next state is being a platooning vehicle, then the vehicle can become either a **Leader** or a **Follower**, depending on the indicated in the configuration file. To become a **Follower**, first, the vehicle sends a join request to the **Leader**; if accepted, it will go to the **Joining** state, where the multiple maneuvers and messages will be exchanged until the maneuver is terminated. Then, the vehicle becomes a **Follower** and will maintain its state, following the **Leader's** orders. The **Follower** can leave the **Follower** state by leaving the platooning and becoming a **Non-Platoon** vehicle.

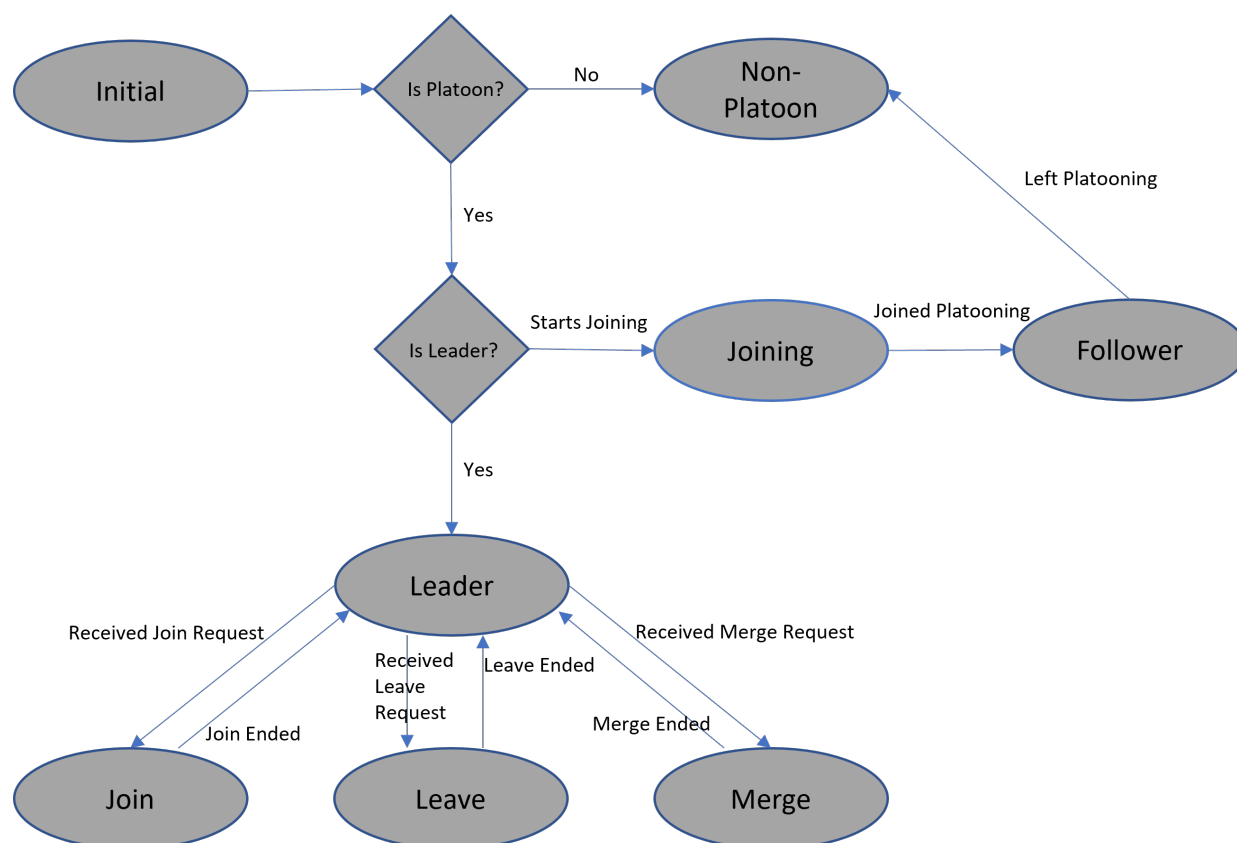


Figure 7.7: Platooning Application State Transition

If the vehicle becomes the Leader, a more complex set of states is available. The first two, **Join** and **Leave**, are triggered by requests sent by the Followers. The **Merge** is a maneuver performed between two Leaders that allows joining two platooning. The **Leader** can only perform one maneuver at each time. It needs to change to a different state until the maneuver is terminated because of the complexity and number of messages and operations needed. For example, in the case of the **Join**, the **Leader** needs to create enough space in the existing platooning to accommodate the new element.

### 7.3.2 Secure Platooning Implementation

The implementation of secure platooning uses the platooning implementation, previously described, as the basis. So, all the communications happen normally, but they are secured using [VPKIbrID](#).

In the secure platooning implementation, the [VPKIbrID](#) is only used to secure the messages. The other entities are used offline, and the vehicles are preloaded with the material they need to communicate securely. The material includes the vehicle [LTC](#) and its [PCs](#), the token, and the [ABE](#) keys for its attributes.

So, using the [VPKIbrID](#) library developed, the first step is to load the material needed for the vehicle.



Then when receiving a message, the first thing the vehicle does is check if the message was directed to it. If so, it uses the algorithms described in the message to select the decryption algorithm. If PKI is indicated it uses its LTC. If encrypted using ABE it uses the corresponding ABE keys. The application can just pass all the parameters to the VPKIbrID library, and it will automatically decrypt and verify the message integrity and authentication.

Implementing the secure platooning in the simulation was not taken further because, after some testing, it was noticed that the implementation of security measures did not affect the results taken from the simulation. It means that the time taken for the cryptographic operations is not taken into account in the simulation. The simulation simply stops until the operations are completed.

## 7.4 Agnostic Middleware for VANETs

Platooning is an application that can run in any vehicle entirely independently of the type of vehicle and its components. However, it needs to access the vehicle internals and communication modules to obtain the information needed for the platooning group (speed, position, etc.) and communicate with the rest of the vehicles. As previously stated, there are multiple approaches to middleware agnosticism (Section 2.2.2); these should give vehicular application developers the freedom to focus on optimizing its components of modules. However, the standard architectures present their shortcomings, either by not presenting solutions for the implementation of simultaneous use of multiple applications on the same ITS station (CALM) or needing knowledge of the lower layers details (ETSI ITS-G5).

The Agnostic and Modular Architecture for the Development of Cooperative ITS Applications [17] is an architecture adapted from the modern standards ETSI and ISO to be deployed on ITS to overcome the presented limitations. It enables the usage of different communication technologies and network/transport protocol stacks, facilitating the creation of ITS cooperative applications and services, allowing applications to use the medium transparently. This architecture is implemented directly on the OBU, which should provide a group of services for access to the vehicle's internal data sources, V2X communication mechanisms, and lower-level functions, identified as information services, communication services, and function services, respectively. Access to the services is done through a common interface to all the services called ITS Local Communication Interface (ITS-LCI). The architecture is shown in Figure 7.8.

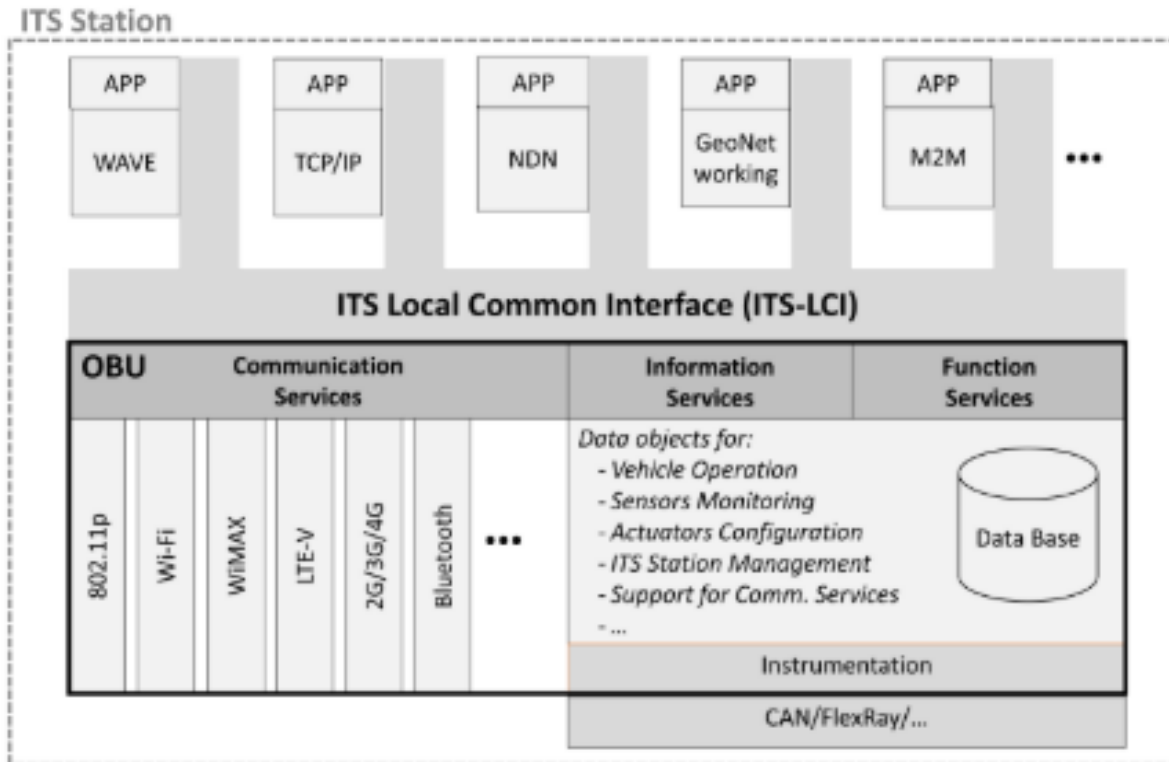


Figure 7.8: Agnostic and Modular Architecture for the Development of Cooperative ITS Applications (From [17])

The architecture was implemented in multiple OBUs from different brands, including Pangea and the one produced by Bosch. It was coded in C++ and implemented the basic services needed by the platooning application.

The communication services provided multiple operations, such as sending and receiving messages from an application connected through the ITS-LCI. It was implemented using the UDP protocol, decreasing the overhead. The communications between the application and the OBU were made through a reliable communication medium, thus not needing the added functionalities provided by Transport Control Protocol (TCP).

The implementation of the information services collected the vehicle data from the CAN and stored it in a Management Information Bases (MIB). Then, the platooning application was able to retrieve the data using the SNMP protocol. The vehicle data included the vehicle's speed, location, and acceleration.

For the function service, no vehicle internals was accessed due to the complexity and legal implications. However, the application allowed to change multiple radio parameters.

The access between the application and the OBU was made through an ethernet cable, providing low latencies and high speeds.

## 7.5 Implementation of CAM Generation Application for Message Collection

The work presented in this Section aims to build large and synthetic datasets containing normal and attacking messages. As presented in Table 2.1, there is a large number of different known attacks in VANETs. This number is constantly growing, so it is necessary to select a subset to be simulated. DoS and fabrication attacks were the selected ones. These were chosen by their simplicity in implementation, facilitating their simulation.

DoS attack tries to disrupt the communications of other vehicles by either saturating the medium or stop the radio equipment to be able to process them all. In the fabrication attack, the attacker vehicle will send messages complying with the rules presented in Table 5.4. But, in this attack, one of the message fields will be replaced by bogus information.

To increase the randomness, each vehicle entering the simulation will have a 10% chance of becoming an attacker instead of having a fixed number of attackers. So, every time a vehicle enters the simulation, it will generate a random number between 0 and 100. If this is smaller or equal to 10, it will become an attacker. Otherwise, it is a normal vehicle. The process is more clearly indicated in Figure 7.9.

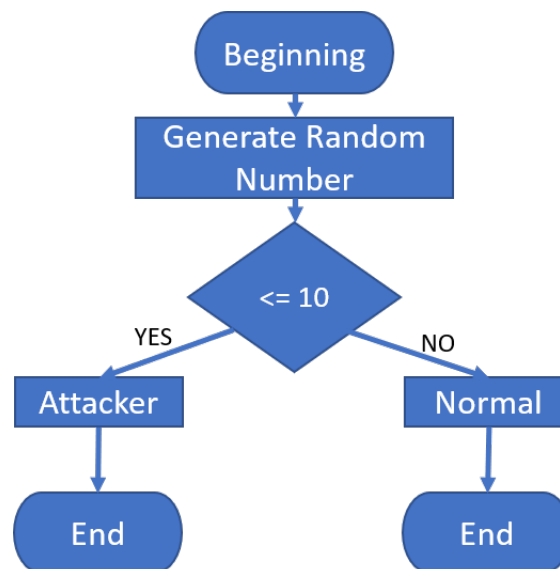


Figure 7.9: Attacker Selection Flow for Message Generation

### 7.5.1 Denial of Service Attack

DoS is a type of attack on availability (Table 2.1). Its goal is to affect the availability of the targeted entities, usually by sending messages at such a rate that the receiving device is occupied and cannot receive authentic messages. In this case, the usage of cryptographic methods, such as signatures, is of little help and can even worsen the problem [8]. Thus, an IDS could be especially helpful in this case, enabling the detection of an attacker without the need to verify signatures.

The vehicles that were previously selected as attackers will attack at random intervals. The attacks will have a random duration that can vary from 0 to 30 seconds. The selection of the attack time and duration is presented in Figure 7.10. If the vehicle is not in attacking mode, it will select a random number from 0 to 100. If this is smaller or equal to 1, it will start the attack, giving it a probability of 1% of starting an attack. Then, it will generate a random number from 0 to 30, serving as the attack duration.

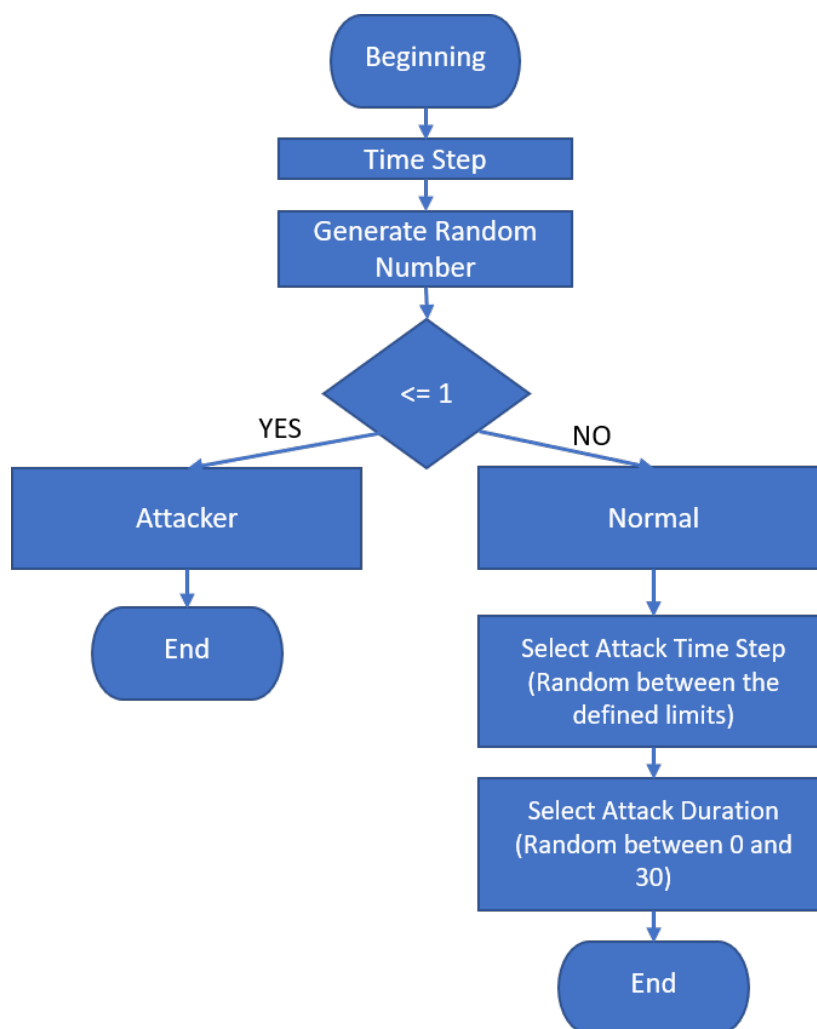


Figure 7.10: Selection of DoS Time Interval and Duration for Message Generation

During the attack period, the attacker will send completely normal messages but at much higher rates. The frequency of the messages sent is also random. It is not possible to directly define the frequency of the messages sent in the simulator, only the time to the next event (the period). So, what is done is to divide the minimum time step used to generate CAMs using the following formula:  $new\_time\_step = ((interval)/100) * old\_time\_step$ . *new\_time\_step* refers to the new calculated time step, *interval* is the interval of multiple values that can be used to calculate the new time step. This work uses the intervals 1 to 10, 10 to 20 and, 20 to 30. Finally, the *old\_time\_step* represents the predefined time step, usually 100 ms (the minimum for a CAM). So, for example, if the interval selected is the first (1 to 10), the new time step is between 1ms and 10 ms.

### 7.5.2 Fabrication Attack

The Fabrication Attack is an attack on integrity and data trust. It may be carried in multiple ways, as, for example, the attacker may try to create a false message or merely modify data on an authentic message. The goal of these types of attacks may be to gain the trust of entities, create a false trail or simply affect the target system. For example, in the case of VANETs, a rogue vehicle may try to create an accident by sending false information about its speed or location. So, contrary to the DoS attack, the Fabrication Attack implemented in this work will follow the rules presented in Table 5.4, but this time the data in the CAM will convey fake data. The attacker will simulate a rogue application or faulty sensor that, at a random time, will produce fake data produced by authentic vehicles.

The attackers and the attack interval will have the same selection process as in the DoS. From the fields existing in the CAM, three were chosen to be faked: Speed, Heading, and Acceleration. The field to be changed is selected previously in the simulation. At each time step, if in attack mode, the vehicle will generate a random value and use it instead of the original vehicle. The fake values should be in the range of the normal ones, hence being easily passable as normal ones. The range of the possible values is calculated by running simulations without attackers. The range of the values for the vehicles are: speed varies from 0 to 14 m/s, heading varies from 0 to 360°, and acceleration from -40 to 11 m/s<sup>2</sup>.

## 7.6 Implementation of Machine Learning Algorithms

The IDS proposed in this research uses ML algorithms to enhance its detection capabilities. It uses the generated datasets to analyze the data exchanged in the network and detect attacks. In this work, ML has been used at two levels; the first was to evaluate the behavior of these algorithms depending on the data fed and, simultaneously, the data itself. Then, use the data from multiple sources to train an ML algorithm to detect attacks in another source at multiple network levels. Both approaches are described next.

### 7.6.1 Implementation of ML Algorithms for Dataset Evaluation

The evaluation of the datasets using ML algorithms demanded a tool flexible enough to change the algorithm parameters between tests. Also, it should provide mechanisms to load one or multiple datasets either for testing and training. Thus, enabling testing the influence of the multiple datasets on the detection.

The first approach tried was to develop an ML tool that was able to receive one or multiple datasets to be trained. This approach was taken due to the flexibility it provides. One of the most used languages in the development of ML tools is Python. There is a significant quantity of documentation available and multiple libraries with a vast community updating them.

The development of the ML tool used the widely available library Scikit-learn. It implements multiple ML algorithms and provides easy development of applications. The development resulted in a web-based platform that allowed the feeding of multiple datasets for training, facilitating the usage of datasets from different sources. It implemented NN, Random Forrest, and SVM. The web GUI is divided into 2 parts. The first represents the processing part, where the user can indicate the ML algorithms, their parameters, and the datasets that will be used (Figure 7.11). The second part shows the results of the evaluation (Figure 7.12).

## DATA

Training Files			Test Files	
Selected Dir: /home/fbgoncalves/datasets/			Selected Dir: /home/fbgoncalves/datasets/	
<b>Directories</b>	<b>File</b>	<b>Select</b>	<b>Selected</b>	<b>Selected</b>
...	dos_0-10_1.csv	<input type="checkbox"/>	...	dos_0-10_1.csv
dos_datasets	dos_test_1.csv	<input type="checkbox"/>	dos_datasets	dos_test_1.csv
header	out_7_acc.csv	<input type="checkbox"/>	header	out_7_acc.csv
no_attacks	out_7_heading.csv	<input type="checkbox"/>	no_attacks	out_7_heading.csv
random_info_datasets	out_7_speed.csv	<input type="checkbox"/>	random_info_datasets	out_7_speed.csv
random_info_maps	random_acc_3.csv	<input type="checkbox"/>	random_info_maps	random_acc_3.csv
test	random_acc_3_test.csv	<input type="checkbox"/>	test	random_acc_3_test.csv

### Engine Configuration

**Names**

recv\_veh,send\_veh,recv\_time,diff\_time,heading,speed,long\_acc,gen\_time,elevation,latitude,longitude,bit\_len,diffheading,diffPos,diffElevation,diffSpeed,diffAccel,is\_attacker,

**Filters**

recv\_veh,send\_veh,recv\_time,heading,speed,long\_acc,longitude,latitude,elevation,gen\_time,bit\_len,

ML -Algorithm NN

Learning Rate Adaptive

Solver SGD

Activation Logistic

Layers 2 4 2

Tol 0.0001

Learning rate init 0.1

Iteration no change 5

Epochs 20

Batch Size 64

Random State 1

## Not Processing

Figure 7.11: Web-based GUI for ML Evaluation - Processing

Dataset Data					Classification Data	
Total Test	Total Positives	Total Negatives	Perc Positives	Perc Negatives	Corr Classified	Corr Positives
0	0	0	0	0	0	0
Total Train	Total Positives	Total Negatives	Perc Positives	Perc Negatives	Incor Classified	Cor Negatives
0	0	0	0	0	0	0

Metrics		Classification Report				Confusion Matrix	
Accuracy	Precision	Class 0				TP	FP
0	0	F1	Precision	Recall	Support	0	0
MEAE	RMSE	0	0	0	0	FN	TN
0	0	F1	Precision	Recall	Support	0	0
Recall		0	0	0	0		
0							

Refresh

Clear

☐ Perform evaluation in multiple test datasets

Submit

Figure 7.12: Web-based GUI for ML Evaluation - Results



## 7.6.2 Hierarchical Intelligent IDS ML Algorithms Implementation

However, despite the advantages of the flexibility provided by the implementation of the Python tool, the effort needed for the implementation of multiple ML algorithms limits the number of algorithms that were possible to train. So, another solution was to use a platform with already implemented algorithms, in which a few clicks were enough to change the algorithms used.

Weka is a standalone application with multiple ML algorithms, thus providing an easier way to test and train multiple algorithms. It provides a command-line tool and a graphical interface. Weka has several tools that allow multiple operations, such as filter data in the dataset, analyze the dataset's parameters, and use labeled and unlabeled learning. Moreover, Weka provides vast information on the loaded data, including the average value, the standard deviation, and the number of samples.

Weka provides a great variety of ML learning algorithms, easily allowing multiple tests to be made quickly. It accepts the datasets using a proprietary format called "ARFF," which is very similar to a "CSV" file, but it includes a header describing the data.

Additionally to the graphical interface, Weka also provides a Java library that can easily be added to code development. So, it provides a good solution to test multiple types of algorithms using the graphical interface and, after the selection of the best solution, use the library for its implementation in a real-world solution.

**Loading Datasets** As previously mentioned, Weka provides a well-documented library. It is made available through Maven [109], facilitating its usage. The first step needed to classify the data and common to all the algorithms is Loading the datasets, training, and testing. Figure 7.13 shows the process to do so.

```
DataSource trainDataset = new DataSource("train.arff");
DataSource testDataset = new DataSource("test.arff");

Instances train = trainDataset.getDataSet();
Instances test = testDataset.getDataSet();
```

Figure 7.13: Loading Dataset Using Weka Java Library

Firstly the raw data is load from a file to a DataSource object, and then the instances are extracted. However, the dataset needs to be treated for classification by indicating the index of the class attribute, shown in Figure 7.14, and then removing the attributes in excess. The removal of the datasets is done by creating a filter, as shown in Figure 7.15; the filter will then be applied during the classification.

```
train.setClassIndex(17);
```

Figure 7.14: Setting Classifier Index Using Weka Java Library

```

Remove rm = new Remove();
rm.setAttributeIndices("1,2,6,7,8,9,10,11,12");
FilteredClassifier classifier = new FilteredClassifier();
classifier.setFilter();

```

Figure 7.15: Adding Filter to the Classifier Using Weka Java Library

**Random Forrest Algorithm Implementation** After loading the datasets and transforming them into instances, the data can now be used to train the model. The algorithm used in this case is [RF](#). First, the [RF](#) object is created, and then the training instances and the filter are loaded, as shown in [Figure 7.16](#). The trained model can then be used to classify the instances in the test datasets, as shown in [Figure 7.17](#).

```

RandomForest rf = new Random();
classifier.setClassifier(rf);
classifier.buildClassifier(train);

```

Figure 7.16: Building Model With [RF](#) Using Weka Java Library

```

for(int i = 0; i < test.numInstances(); i++){
    double pred = classifier.classifyInstance(test.instance(i));
}

```

Figure 7.17: Classifying Test Instances Using Weka Java Library

**Ensemble Learning Implementation** The process of building the ensemble model is a little more complex. It follows the same steps, but it needs to build the multiple models before building the ensemble as the method **addPreBuiltClassifiers**, only accepts classifiers that have already been built. The process is shown in [Figure 7.18](#). First, the multiple classifiers are individually built. Then, these are joined together into a "Vote" classifier. The latter will be used to classify the data.

```

Vote vote = new Vote ();
RandomForest randomForest = new RandomForest ();
randomForest.buildClassifier (train );

MultilayerPerceptron multilayerPerceptron = new MultilayerPerceptron ();
multilayerPerceptron.buildClassifier (train );

J48 j48 = new J48 ();
j48.buildClassifier (train );

vote.addPreBuiltClassifier (randomForest );
vote.addPreBuiltClassifier (multilayerPerceptron );
vote.addPreBuiltClassifier (j48 );
vote.buildClassifier (train );

```

Figure 7.18: Building Ensemble Voting Using Weka Java Library

**Rule-Based Learning Implementation** As for Rule-Based Learning, the method for creating and training the model is the same as for the [RF](#). However, this algorithm creates a one-step decision tree, meaning that the rules can be written as an if/else clause, which is lighter and smaller to be transmitted over the network. In this case, multiple rules were created, one for each attack, [DoS](#), and the 3 different fabrication attacks. The if clauses created are shown in [Figure 7.19](#). So, when a vehicle receives a message, it will verify if the parameter calculated as `diff_x` falls between the threshold. If not, the message is considered an attack.

```

if (diff_time < threshold){
    attack=true;
}
if (threshold_min < diff_speed < threshold_max){
    attack=true;
}
if (threshold_min < diff_acc < threshold_max){
    attack=true;
}
if (threshold_min < diff_heading < threshold_max){
    attack=true;
}

```

Figure 7.19: Building Ensemble Voting Using Weka Java Library

## 7.7 Real World Platooning Implementation

The platooning application and its underlying protocol were researched until it was ready for a real-world application, with the implementation in the simulation environment being of key importance.

The real-world implementation of the platooning application takes advantage of the agnostic architecture described in Section 7.4. It allows the separation of software development and vehicle internals by providing an easy interface for communicating with others and accessing the necessary vehicle parameters. The application, previously developed in Java, was ported to android. Android tablets are portable and easy to use in the vehicle. The applications use the [ITS-LCI](#) provided by the agnostic architecture to access the vehicle's speed, acceleration, and [GPS](#).

The tests were run on the campus of the University of Minho using two different vehicles. First, the Leader starts its application, broadcasting its existence and position. Then, after receiving the beacon announcing the platooning position, the Follower approaches and requests to Join. The Follower Joins the platooning, follows the Leader, and finally requests to Leave. The complete path traveled by the vehicles is presented in Figure 7.20. All the maneuvers and vehicle driving are made manually due to the vehicle's technology limitations and the embryonic state of the application.



Figure 7.20: Platooning Real-World Implementation Circuit

The setup is shown in Figure 7.21. The vehicle provides access to its data through its Controller Area Network ([CAN](#)). The access is performed using an [OBU](#) connected to a wireless router. The application can connect to the [OBU](#) and use the [ITS-LCI](#) to use the [OBU](#) communications and access the vehicle data.

Figure 7.22 shows the laboratorial demo performed. The top two screens are screenshots of the android



Figure 7.21: Platooning Real-World Implementation Setup

platooning application. The left one is the application screen from the Leader's standpoint. It shows the multiple vehicles of the platooning, representing the Follower in red and the Leader in blue. It also shows the multiple parameters of each vehicle, including their identification and speed. The right screen shows the Follower application. It shows the speed of each vehicle and its position on the map. Furthermore, the Follower has the option to Leave the platooning.

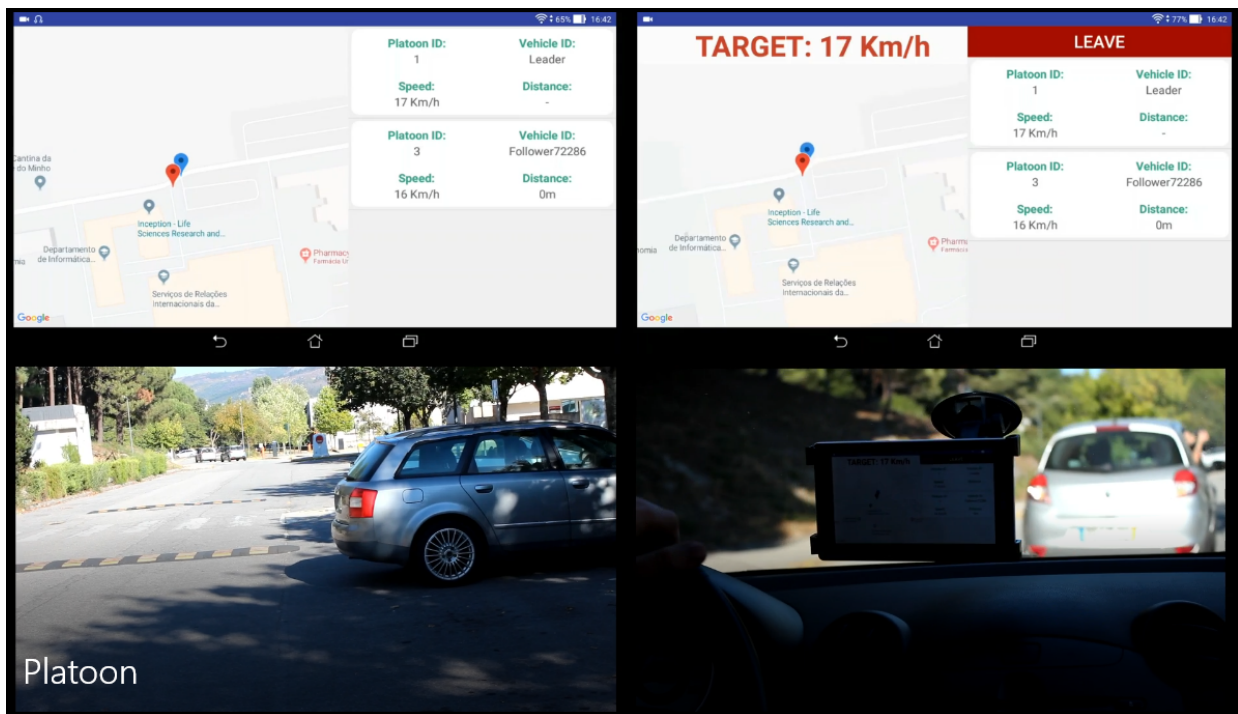


Figure 7.22: Platooning Real-World Implementation Application Example and Scenario

# Chapter 8

## Test and Evaluation Procedures

This Chapter presents the outcomes resulting from the multiple new protocols and methodologies proposed in this thesis work. These enable the evaluation of the presented work and comparison with other future and existing works.

Firstly, it is presented the performance evaluation results for the [VPKIbrID](#), comparing its multiple modes and ciphers, providing insight on which are better suited for each situation. Then, detailed information about the collected datasets is presented, including how each parameter collected varies in each simulation, the number of normal and attacker vehicles. Next, the datasets are fed to an [ML](#) algorithm to evaluate the possibility of detecting an attack on them, their similarity, and their quality. Finally, the datasets collected are used to test and train multiple [ML](#) algorithms. The results should provide some insight into the better-suited algorithms for each hierarchy level, balancing accuracy with [CPU](#) needs and detection time.

### 8.1 Evaluation of Secure Communications

[VPKIbrID](#) is an application layer security model that implements multiple technologies to respond to the needs of [VANET](#) applications. It is based on two cryptographic technologies, namely, [PKI](#) and [ABE](#). However, cryptographic operations are computationally demanding, affecting the usual behavior of the system by introducing delays, overhead, and computational power needs.

So, a Java implementation was made to evaluate the impact of the security model on the overall communications using Java JDK 8. The pairing operations needed by the [ABE](#) were implemented using the [JPBC](#). For the [ABE](#), an existing GitHub implementation was used (<https://github.com/junwei-wang/cpabe>). This implementation was not ideal since it uses type 1 pairings (easier to implement but with worst performance). The [JPBC](#) library allows the usage of a C wrapper that provides better performance. The application was run

a total of 100 000 times for each configuration using a computer with an Intel Core i7 and 16 GigaBytes of RAM.

VPKIbrID was tested using the three following configurations:

1. Public Key Infrastructure (PKI)
2. Attribute-Based Encryption (ABE) with Message Authentication Code (MAC) signatures.
3. Attribute-Based Encryption (ABE) with Public Key Infrastructure (PKI) signatures.

However, the library provides a C wrapper so, the three different configurations were also tested with and without the wrapper. Moreover, ABE cryptography is inherently slow and CPU-heavy. So a mechanism was implemented that allows the implemented VPKIbrID to cache the used keys. This mechanism bypasses the generation of the "Randomly generated key" (Figure 4.4) if multiple messages are encrypted using the same attributes for the same targets. Using this mechanism, if multiple messages are being sent for the same targets using the same attributes, VPKIbrID-ABE will only need to encrypt the information using the previously generated symmetric key, increasing its performance.

Tables 8.1 and 8.2 present the results obtained for the encryption and decryption operations, respectively. Each column represents a VPKIbrID operation. The last two columns refer to the total encryption (TE) and total decryption (TD) times, respectively. Each line represents a different configuration. From the results obtained, possible outliers were removed. Due to some factors external to the tests, sometimes processing time is given to other processes, which influences the performance measures. This impact was most noticeable in operations with smaller values in, for instance, the key generation.

Tables 8.1 and 8.2 are also divided into 3 sub-tables. The first one contains results obtained for the operations without the C wrapper. The second, the ones using the wrapper and, the final part, the results using the wrapper and key caching. VPKIbrID-PKI results are only in the first Table. The reason is that the wrapper and key cache do not make any difference in this case.

As expected, in the results for the "no C wrapper configuration", VPKIbrID-PKI had a better performance than the others. ABE encryption is computationally expensive. The slowest configurations use VPKIbrID-ABE with public key signatures. The usage of a MAC algorithm improved the total encryption time by about 6 milliseconds, which is the time taken by signing a message with a public key.

The second part of the Table contains the results obtained with the C wrapper. The obtained results were not very significant. The difference was less than a millisecond for the total encryption and decryption times.

Finally, the results using ABE key caching are presented in the last part of the Table. These present a great improvement over the other results. In type 2 configurations, the total encryption and decryption take less



Table 8.1: Comparison Between VPKIbrID Modes - Encryption Performance

Without PBC wrapper

Type	SK gen (ms)	DE (ms)	DS (ms)	KE (ms)	TE (ms)
1	< 0.02	< 0.11	$5.88 \pm 0.10$	< 0.30	$6.13 \pm 0.11$
2	< 0.01	< 0.08	< 0.01	$106.84 \pm 1.21$	$106.93 \pm 1.21$
3	< 0.03	< 0.08	$5.79 \pm 0.06$	$108.89 \pm 1.15$	$112.79 \pm 1.16$

With PBC wrapper

Type	SK gen (ms)	DE (ms)	DS (ms)	KE (ms)	TE (ms)
2	< 0.01	< 0.08	< 0.01	$105.65 \pm 1.19$	$105.74 \pm 1.19$
3	< 0.03	< 0.08	$5.92 \pm 0.08$	$105.95 \pm 1.16$	$111.97 \pm 1.17$

With ABE cached keys

Type	SK gen (ms)	DE (ms)	DS (ms)	KE (ms)	TE (ms)
2	< 0.01	< 0.06	< 0.01	< 0.01	< 0.08
3	< 0.01	< 0.12	$5.81 \pm 0.08$	< 0.01	$5.87 \pm 0.89$

SK gen - Symmetric Key generation; DE - Data Encryption; DS - Data Signing; KE - Key Encryption; TE - Total Encryption;

Table 8.2: Comparison Between VPKIbrID Modes - Decryption Performance

Without PBC Wrapper

Type	KD (ms)	SV (ms)	DD (ms)	TD (ms)
1	$5.73 \pm 0.10$	< 0.23	< 0.12	$5.98 \pm 0.11$
2	$36.22 \pm 0.45$	< 0.01	< 0.10	$36.35 \pm 0.45$
3	$36.20 \pm 0.36$	< 0.20	< 0.10	$36.51 \pm 0.36$

With PBC Wrapper

Type	KD (ms)	SV (ms)	DD (ms)	TD (ms)
2	$36.42 \pm 0.55$	< 0.02	< 0.10	$36.55 \pm 0.55$
3	$36.55 \pm 0.40$	< 0.21	< 0.11	$36.86 \pm 0.40$

With ABE cached keys

Type	KD (ms)	SV (ms)	DD (ms)	TD (ms)
2	< 0.01	< 0.01	< 0.03	< 0.18
3	< 0.01	< 0.23	< 0.11	$0.23 \pm 0.03$

KD - Key Decryption; SV - Signature Verification; DD - Data Decryption; TD - Total Decryption;

than a millisecond. In type 3, the elapsed time for the total encryption increases to less than 7 milliseconds due to the public key signing process.

Due to the added cryptography, it is expected for overhead to be introduced, increasing message size. Table 8.3 compares the overhead introduced by the 3 encryption modes **PKI**, **ABE** with **MAC** signatures, and **ABE** with **PKI** signatures. This Table was constructed by encrypting messages with different sizes: 128, 256, 512, 1024, and 2048 bytes. The values indicated in each column are the following: Plain Message - the size of the payload data; **VPKIbrID-PM** - the size of the total message after constructing the **VPKIbrID-PM**; Encrypted Data: the size of the encrypted **VPKIbrID-PM**; Encrypted Key: the size of the encrypted key (key used to encrypt the data); **VPKIbrID-EM**: the size of the **VPKIbrID-EM**; Total overhead: the overhead introduced by the cryptographic material.

The obtained results are directly related to the algorithms and ciphers utilized. In this evaluation, the following parameters were used: signatureAlgorithm - SHA256withRSA (**PKI**) and hmacSHA256 (symmetric); encKeyAlgorithm - **ABE** or RSA 2048 for **PKI**; encAlgorithm - AES 128.

Table 8.3: Comparison Between **VPKIbrID** Modes - Message Size (Bytes)

Plain Message	PKI					ABE with MAC					ABE with PKI signatures				
	PM	ED	EK	EM	TO	PM	ED	EK	EM	TO	PM	ED	EK	EM	TO
128	2672	2688	256	3052	2924	224	240	853	1205	1077	2672	2688	853	3654	3562
256	2801	2816	256	3180	2924	354	368	853	1334	1078	2801	2816	853	3782	3526
512	3057	3072	256	3436	2924	610	624	853	1590	1078	3057	3072	853	4038	3526
1024	3569	3584	256	3948	2924	1122	1136	853	2102	1078	3569	3584	853	4550	3526
2048	4593	4608	256	4972	2924	2146	2160	853	3126	1078	4593	4608	853	5574	3526

PM - **VPKIbrID-PM**; ED - Encrypted Data; EK - Encrypted Key; EM - **VPKIbrID-EM**; TO - Total Overhead;

It is possible to see that the total overhead introduced is constant and independent of the original payload. The main factor in the increased message size is the need to send the sender certificate in the modes with **PKI** signatures. Thus, being the reason for the **ABE** with **MAC** signatures to be the least affected by the overhead (1062 bytes), although being the one with the biggest resulting encrypted key. The **ABE** mode with **PKI** signatures is affected by the drawbacks of both methods, being the one with the biggest overhead (3562 bytes).

Concluding, the **ABE** keys generation is extremely expensive (performance-wise). These are only useful in scenarios where multiple target encryption or key caching can be taken advantage of. Also, it can be used if the public key of the receiver is unknown or the message size is of importance. Otherwise, **VPKIbrID-PKI** is more indicated. It is much less expensive, mainly when there is only one receiver. The main drawback is the need to know the public key of the receiver and increased message size.

## 8.2 Evaluating the Impact of Biased Datasets

One of the first steps of this thesis work was an [SLR](#) (Chapter 3). It provided insightful information about other existing works in the literature in a methodic and systematic way. One of the results was the source and method of creating the datasets in the found research work. Most researchers use datasets made by themselves and do not provide the methodology used or the datasets. So, it is hard to verify and replicate their results and compare obtained results.

So, before designing the methodology to create the datasets, it was decided to perform a test to assess the influence or even the dependency that datasets put on the derived results. It does not try to show how good the classifier is, but how it can produce biased results depending on the dataset.

Using [SUMO](#), a scenario has been built that comprises a 4x4 grid where vehicles follow 10 specific routes. The normal behavior vehicles will broadcast messages in 1-second intervals.

The attackers are selected by generating a random number between 0 and 100. If it is smaller than 10, the vehicle is selected as an attacker. If it is chosen as an attacker, it will randomly start to perform [DoS](#) attacks. It will send the same messages as the normal vehicles, but with a smaller time interval, which will be randomly chosen from 1 to 40% of the normal period. The attacks will be made at random times during a random interval until a max of 60 seconds. The test and training of the [ML](#) algorithms were done using Weka.

All messages received by the vehicles were stored in the same dataset and are marked as attack or normal. From the dataset, the top 200,000 messages were chosen to train a classifier. These were divided into training (80%) and test (20%). This division is made randomly using Weka. [MLP](#) was used to classify the collected training set using 10 fold cross-validation. The algorithm configuration is the default.

The results of training phase have 100% accuracy, which indicates possible overfitting. The trained model is then used to classify the test dataset. The accuracy of the classifier for this dataset is 100%, with all messages well classified.

Then, the same scenario has been run again to test how much the results may be biased, creating a completely new dataset. As the chosen attacker will be random and the attack intervals too, this dataset will have completely different attacker vehicles with different [DoS](#) attacks. The previously trained model is then used to predict the results in the new datasets. The detection rate is much lower, decreasing to 67%. The [TPR](#) for the abnormal messages is only 55%, meaning almost half of the attacks were considered normal messages.

The presented results do not mean that the results of the found studies are biased, but it is impossible to verify the results obtained without the datasets used to train and test the models.

## 8.3 Evaluation of the Collected VANET Datasets

In Chapter 5, a methodology to collect messages was presented. It simulated vehicles in 7 different geographical maps that used VANET communications to disseminate messages. The goal was to produce datasets to train and test IDSs for VANETs. So, these datasets also include attacks, more precisely, DoS and fabrication attacks. The latter has multiple variations; in each, the attacker can fabricate one of the parameters, speed, acceleration, or heading.

The results of evaluating the datasets will be presented at two levels: First, the statistical analysis of the contents of the datasets, including the number of vehicles in each simulation, the number of attackers, the number of exchanged messages and attacks. Then, the datasets are fed to an ML learning algorithm to gauge the possibility of detecting attacks on the datasets, the quality of the datasets and their similarity, the advantage of having multiple datasets and the behavior of the IDS if located at different places of the network.

### 8.3.1 Evaluation of the Collected Datasets

This Section presents and analyses the content of the datasets, for example, the number of vehicles in each simulation, the number of attackers, the number of exchanged messages and attacks.

Firstly are presented the results for the DoS attacks. These can be seen in Tables 8.4, 8.5, and 8.6. Tables 8.7, 8.8 and 8.9, present the results for the random heading, speed, and acceleration, respectively.

The presented tables are composed of two main parts. The first presents the dataset composition, the number of normal and attack messages, the size of the dataset, and the number of normal and attacker vehicles. The second block is divided into two, normal (first part) and attack (second part) messages. These present the average and standard deviation of the data contained in the received messages, including the `diffSpeed`, **`diffAcceleration`**, **`diffHeading`**, and **`diffTime`**. All these values represent the difference between the values received in two consecutive received messages from the same vehicle. For example, if V0 was received in the instant T0 and V1 was received in the instant T1. T0 is the instant when a CAM was received from vehicle 0, and T1 is the instant when the next message was received. The value in the Table is the average of V0 and V1. More generically, these values are the average of the several  $V_t$  and  $V_{(t+1)}$ . The same happens for the values of heading, acceleration, and time.

Each line of the column represents the values obtained for each map in Table 5.1. All the datasets and code are available for evaluation and further use in github <https://github.com/fabio-r-goncalves/dataset-collection>.

The values presented in Tables 8.4, 8.5, and 8.6 refer to the first scenario, where the attackers perform DoS. The most noticeable difference between the normal and attack values is the `diffTime`. The average value

Table 8.4: Collected DoS Dataset Results - Interval 1 - 10%

Map	Positives				Negatives				Normal				Attack			
	Positives	Negatives	Total	FS (mb)	Attackers	Normal	Total	FS (mb)	diffSpeed	diffAcceleration	diffHeading	diffTime	diffSpeed	diffAcceleration	diffHeading	diffTime
1	42264 (64.57%)	23193 (35.43%)	65457	12.4	1 (3%)	29 (97%)	30	12.4	0.01 ± 0.45	0.02 ± 10.65	-0.33 ± 27.13	327 ± 161	0.00 ± 0.00	0.00 ± 0.13	0.00 ± 0.14	4 ± 5
2	127946 (65.96%)	100707 (44.04%)	228653	43.7	2 (7%)	28 (93%)	30	43.7	-0.01 ± 0.52	-0.01 ± 7.24	0.49 ± 24.44	433 ± 766	0.00 ± 0.04	0.00 ± 0.80	0.04 ± 10.64	10 ± 161
3	178999 (76.29%)	55621 (23.71%)	234620	44.7	2 (10%)	18 (90%)	20	44.7	0.01 ± 0.61	0.02 ± 9.39	0.31 ± 32.36	379 ± 261	0.00 ± 0.04	0.00 ± 3.08	0.01 ± 0.62	7 ± 15
4	815990 (72.85%)	304170 (27.15%)	1120160	213.0	6 (10%)	57 (90%)	63	213.0	0.00 ± 0.47	0.00 ± 8.18	0.26 ± 32.42	426 ± 2987	0.00 ± 0.07	0.00 ± 2.70	0.01 ± 11.77	8 ± 121
5	150539 (65.33%)	79893 (34.67%)	230432	44.3	2 (7%)	25 (93%)	27	44.3	0.00 ± 0.34	0.01 ± 7.08	-0.18 ± 19.35	363 ± 588	0.00 ± 0.05	0.00 ± 3.46	0.00 ± 0.70	7 ± 14
6	330231 (79.91%)	83000 (20.09%)	413231	78.9	3 (10%)	27 (90%)	30	78.9	-0.02 ± 0.51	0.02 ± 8.39	0.23 ± 26.38	356 ± 1684	0.00 ± 0.06	0.00 ± 2.73	0.02 ± 10.10	6 ± 222
7	7756817 (83.30%)	1555401 (16.70%)	9312218	1730.0	19 (11%)	157 (89%)	176	1730.0	0.01 ± 0.53	0.01 ± 7.07	0.22 ± 40.90	1218 ± 3996	0.00 ± 0.04	0.00 ± 2.45	0.01 ± 8.44	21 ± 483

Table 8.5: Collected DoS Dataset Results - Interval 10 - 20%

Map	Positives				Negatives				Normal				Attack			
	Positives	Negatives	Total	FS (mb)	Attackers	Normal	Total	FS (mb)	diffSpeed	diffAcceleration	diffHeading	diffTime	diffSpeed	diffAcceleration	diffHeading	diffTime
1	19091 (44.69%)	23624 (55.31%)	42715	8.2	1 (3%)	29 (97%)	30	8.2	0.01 ± 0.47	0.01 ± 10.55	-0.28 ± 26.68	325 ± 158	0.00 ± 0.03	0.01 ± 2.34	-0.04 ± 8.24	3 ± 10
2	503901 (84.17%)	94804 (16.83%)	598705	108.0	7 (2%)	28 (98%)	30	108.0	-0.01 ± 0.53	-0.01 ± 7.24	0.61 ± 25.80	451 ± 802	0.00 ± 0.05	0.00 ± 2.48	-0.01 ± 0.97	4 ± 66
3	59688 (50.12%)	59395 (49.88%)	119083	20.9	1 (5%)	19 (95%)	20	20.9	0.01 ± 0.58	0.02 ± 9.26	0.32 ± 31.35	369 ± 236	0.00 ± 0.05	0.00 ± 1.89	-0.02 ± 0.77	7 ± 19
4	975985 (77.97%)	275749 (22.03%)	1251734	239.0	7 (11%)	56 (89%)	63	239.0	0.00 ± 0.52	0.00 ± 8.50	0.25 ± 32.52	461 ± 3006	0.00 ± 0.05	0.00 ± 2.14	0.02 ± 15.12	9 ± 493
5	149850 (64.60%)	82119 (35.40%)	231969	44.3	2 (7%)	25 (93%)	27	44.3	0.00 ± 0.34	0.01 ± 7.27	-0.18 ± 19.16	360 ± 579	0.00 ± 0.02	0.00 ± 1.02	0.00 ± 0.25	4 ± 27
6	176707 (67.26%)	86032 (32.74%)	262739	50.6	2 (7%)	28 (93%)	30	50.6	-0.02 ± 0.51	0.02 ± 8.23	0.29 ± 26.73	353 ± 1706	0.00 ± 0.06	0.00 ± 2.94	-0.01 ± 1.83	6 ± 91
7	7208829 (78.74%)	1946871 (21.26%)	9155700	1690.0	11 (6%)	94 (165%)	176	1690.0	0.01 ± 0.47	0.01 ± 6.81	0.19 ± 37.52	1019 ± 3592	0.00 ± 0.03	0.00 ± 2.46	0.01 ± 10.05	12 ± 357

Table 8.6: Collected DoS Dataset Results - Interval 20 - 30%

Map	Positives				Negatives				Normal				Attack			
	Positives	Negatives	Total	FS (mb)	Attackers	Normal	Total	FS (mb)	diffSpeed	diffAcceleration	diffHeading	diffTime	diffSpeed	diffAcceleration	diffHeading	diffTime
1	26891 (53.80%)	23089 (46.20%)	49980	9.8	1 (3%)	29 (97%)	30	9.8	0.01 ± 0.47	0.01 ± 10.57	-0.34 ± 27.18	326 ± 157	0.00 ± 0.08	0.01 ± 3.30	0.01 ± 0.86	8 ± 10
2	622784 (89.38%)	73981 (10.62%)	696765	131.0	6 (20%)	24 (80%)	30	131.0	-0.01 ± 0.64	-0.01 ± 7.83	0.71 ± 26.75	531 ± 940	0.00 ± 0.06	0.00 ± 2.98	0.01 ± 6.39	9 ± 95
3	443652 (90.30%)	47644 (9.70%)	491296	91.7	3 (15%)	17 (85%)	20	91.7	0.01 ± 0.67	0.01 ± 9.40	0.30 ± 31.77	421 ± 313	0.00 ± 0.07	0.00 ± 2.99	0.01 ± 14.52	5 ± 12
4	1356717 (84.22%)	254246 (15.78%)	1610963	308.0	7 (11%)	56 (89%)	63	308.0	0.01 ± 0.55	0.01 ± 8.63	0.31 ± 32.95	485 ± 3116	0.00 ± 0.05	0.00 ± 2.80	0.00 ± 10.81	9 ± 444
5	617909 (91.39%)	58240 (8.61%)	676149	18.0	8 (30%)	19 (70%)	27	18.0	0.00 ± 0.41	0.00 ± 7.63	-0.26 ± 22.38	423 ± 693	0.00 ± 0.03	0.00 ± 2.36	0.00 ± 2.28	9 ± 53
6	409271 (85.52%)	69285 (14.48%)	478556	91.7	6 (20%)	24 (80%)	30	91.7	-0.01 ± 0.58	0.03 ± 8.38	0.30 ± 27.26	393 ± 1717	0.00 ± 0.07	0.00 ± 3.22	0.01 ± 9.82	10 ± 353
7	8023432 (82.03%)	1757105 (17.97%)	9780537	1820.0	17 (10%)	159 (90%)	176	1820.0	0.01 ± 0.50	0.01 ± 6.85	0.24 ± 39.20	1108 ± 3764	0.00 ± 0.03	0.00 ± 2.22	0.00 ± 7.53	14 ± 411

Table 8.7: Collected Dataset Results - Fake Heading Attack

Map	Positives	Negatives	Total	FS (mb)	Attackers	Normal	Total	Normal				Attack			
								diffSpeed	diffAcceleration	diffHeading	diffTime	diffSpeed	diffAcceleration	diffHeading	diffTime
1	1315 (5.53%)	22476 (94.47%)	23791	4.7	3 (10%)	27 (90%)	30	0.01 $\pm$ 0.47	0.01 $\pm$ 10.56	-0.31 $\pm$ 27.65	324 $\pm$ 157	0.01 $\pm$ 0.48	0.19 $\pm$ 9.74	-1.47 $\pm$ 155.75	340 $\pm$ 155
2	3655 (3.48%)	101500 (96.52%)	105155	20.1	2 (7%)	28 (93%)	30	-0.01 $\pm$ 0.50	-0.02 $\pm$ 7.15	0.49 $\pm$ 24.77	428 $\pm$ 777	-0.06 $\pm$ 0.60	0.21 $\pm$ 5.64	1.07 $\pm$ 137.38	406 $\pm$ 405
3	4415 (7.28%)	56210 (92.72%)	60625	11.7	2 (10%)	18 (90%)	20	0.00 $\pm$ 0.58	0.00 $\pm$ 9.24	0.24 $\pm$ 30.32	370 $\pm$ 238	0.06 $\pm$ 0.56	0.35 $\pm$ 9.07	1.67 $\pm$ 151.03	337 $\pm$ 199
4	9833 (2.78%)	333780 (97.14%)	343613	66.9	3 (5%)	57 (95%)	63	0.00 $\pm$ 0.44	0.00 $\pm$ 7.94	0.13 $\pm$ 31.92	396 $\pm$ 2850	-0.01 $\pm$ 0.44	0.04 $\pm$ 9.64	3.04 $\pm$ 151.22	418 $\pm$ 699
5	3245 (3.84%)	81156 (96.16%)	84401	16.4	2 (7%)	25 (93%)	27	0.00 $\pm$ 0.32	0.00 $\pm$ 7.26	-0.12 $\pm$ 19.15	352 $\pm$ 567	-0.03 $\pm$ 0.49	0.15 $\pm$ 5.90	-0.83 $\pm$ 154.12	467 $\pm$ 652
6	6123 (6.72%)	84955 (93.28%)	91078	17.7	3 (10%)	27 (90%)	30	-0.02 $\pm$ 0.49	0.01 $\pm$ 8.13	0.32 $\pm$ 26.90	345 $\pm$ 1668	0.00 $\pm$ 0.47	0.04 $\pm$ 9.68	-0.03 $\pm$ 147.28	341 $\pm$ 1579
7	183304 (6.94%)	2458792 (93.06%)	2642096	511.0	19 (11%)	156 (89%)	175	0.01 $\pm$ 0.39	0.01 $\pm$ 6.62	-0.01 $\pm$ 34.49	790 $\pm$ 3119	0.01 $\pm$ 0.37	-0.01 $\pm$ 6.71	2.08 $\pm$ 151.88	787 $\pm$ 2896

Table 8.8: Collected Dataset Results - Fake Speed Attack

Map	Positives	Negatives	Total	FS (mb)	Attackers	Normal	Total	Normal				Attack			
								diffSpeed	diffAcceleration	diffHeading	diffTime	diffSpeed	diffAcceleration	diffHeading	diffTime
1	1086 (4.56%)	22705 (95.44%)	23791	4.7	2 (7%)	28 (93%)	30	0.01 $\pm$ 0.50	0.01 $\pm$ 10.60	-0.34 $\pm$ 26.41	326 $\pm$ 156	0.03 $\pm$ 6.36	0.24 $\pm$ 8.58	0.27 $\pm$ 33.60	307 $\pm$ 182
2	7500 (7.13%)	97674 (92.87%)	105174	20.1	3 (10%)	27 (90%)	30	-0.01 $\pm$ 0.57	-0.01 $\pm$ 7.15	0.52 $\pm$ 24.76	424 $\pm$ 772	-0.01 $\pm$ 5.87	-0.04 $\pm$ 6.31	0.43 $\pm$ 20.44	471 $\pm$ 701
3	3725 (6.15%)	56865 (93.85%)	60590	11.7	2 (10%)	18 (90%)	20	0.00 $\pm$ 0.60	0.03 $\pm$ 9.22	0.18 $\pm$ 30.21	365 $\pm$ 233	0.04 $\pm$ 5.39	0.04 $\pm$ 9.42	2.29 $\pm$ 41.34	405 $\pm$ 268
4	25661 (7.46%)	318210 (92.54%)	343871	66.9	7 (11%)	56 (89%)	63	0.00 $\pm$ 0.51	0.00 $\pm$ 8.02	0.27 $\pm$ 31.70	396 $\pm$ 2843	-0.07 $\pm$ 5.55	-0.01 $\pm$ 7.87	-0.36 $\pm$ 34.13	409 $\pm$ 2399
5	2602 (3.08%)	81806 (96.92%)	84408	16.3	1 (4%)	26 (96%)	27	0.00 $\pm$ 0.37	0.1 $\pm$ 7.19	-0.17 $\pm$ 19.19	358 $\pm$ 580	-0.12 $\pm$ 5.85	-0.01 $\pm$ 7.64	-0.17 $\pm$ 2.82	323 $\pm$ 1072
6	7024 (7.71%)	84066 (92.29%)	91090	17.7	3 (10%)	27 (90%)	30	0.01 $\pm$ 0.69	0.01 $\pm$ 8.32	0.17 $\pm$ 26.01	616 $\pm$ 2213	-0.30 $\pm$ 5.93	0.17 $\pm$ 11.62	0.64 $\pm$ 46.33	339 $\pm$ 1587
7	172892 (6.54%)	2469036 (93.46%)	2641928	511.0	17 (9%)	158 (11%)	175	-0.01 $\pm$ 0.52	0.00 $\pm$ 6.62	0.18 $\pm$ 32.60	790 $\pm$ 3096	0.18 $\pm$ 5.69	0.04 $\pm$ 6.81	-0.22 $\pm$ 42.51	778 $\pm$ 3217

Table 8.9: Collected Dataset Results - Fake Acceleration Attack

Map	Positives	Negatives	Total	FS (mb)	Attackers	Normal	Total	Normal				Attack			
								diffSpeed	diffAcceleration	diffHeading	diffTime	diffSpeed	diffAcceleration	diffHeading	diffTime
1	1264 (5.31%)	22527 (94.69%)	23791	4.7	3 (10%)	27 (90%)	30	0.01 $\pm$ 0.47	0.03 $\pm$ 10.72	-0.31 $\pm$ 27.52	324 $\pm$ 158	-0.01 $\pm$ 0.40	-0.38 $\pm$ 21.10	-0.51 $\pm$ 2.94	339 $\pm$ 130
2	3294 (3.13%)	101907 (96.87%)	105201	20.1	2 (7%)	28 (97%)	30	-0.02 $\pm$ 0.50	-0.01 $\pm$ 7.20	0.55 $\pm$ 24.84	428 $\pm$ 775	0.02 $\pm$ 0.52	-0.51 $\pm$ 21.01	-0.61 $\pm$ 6.25	402 $\pm$ 548
3	8623 (14.22%)	52002 (85.78%)	60625	11.7	4 (20%)	16 (80%)	20	0.01 $\pm$ 0.58	0.12 $\pm$ 9.22	0.19 $\pm$ 32.74	367 $\pm$ 238	-0.02 $\pm$ 0.56	-0.57 $\pm$ 21.09	1.05 $\pm$ 17.13	373 $\pm$ 219
4	26286 (7.65%)	317507 (92.35%)	343793	67.0	6 (10%)	57 (90%)	63	0.00 $\pm$ 0.44	0.03 $\pm$ 8.02	0.19 $\pm$ 31.49	396 $\pm$ 2840	0.01 $\pm$ 0.45	-0.41 $\pm$ 20.91	0.56 $\pm$ 36.42	400 $\pm$ 2464
5	7505 (8.90%)	76864 (91.10%)	84369	16.4	4 (15%)	23 (85%)	27	0.00 $\pm$ 0.34	0.05 $\pm$ 7.30	-0.14 $\pm$ 17.93	357 $\pm$ 595	0.02 $\pm$ 0.29	-0.50 $\pm$ 20.95	-0.44 $\pm$ 26.90	350 $\pm$ 197
6	4470 (4.91%)	86641 (95.09%)	91111	17.7	2 (7%)	28 (93%)	30	-0.01 $\pm$ 0.49	0.06 $\pm$ 8.12	0.23 $\pm$ 25.82	345 $\pm$ 1648	-0.07 $\pm$ 0.48	-0.05 $\pm$ 21.96	0.84 $\pm$ 30.94	330 $\pm$ 1910
7	62686 (2.37%)	2578794 (97.63%)	2641480	511.0	9 (5%)	166 (95%)	175	0.01 $\pm$ 0.39	0.03 $\pm$ 6.67	0.14 $\pm$ 33.37	789 $\pm$ 3107	-0.01 $\pm$ 0.37	-0.86 $\pm$ 21.00	0.82 $\pm$ 31.83	812 $\pm$ 2979

of the elapsed time between two consecutively messages is much smaller in the attacks. This result was as expected because the frequency of the attacks is much higher. Also, the number of received messages is much higher for the attacks. Although there are much fewer attackers than normal vehicles, as they generate messages much more quickly, there are more attack messages.

The number of attackers varies from 3 to 30%, and the number of attacking messages from 44 to 91% of the total. The created datasets range from 8.2 to 1820 MB. The last map, represented as the number 7 in all the tables, is bigger and has more vehicles running, originating a bigger dataset.

In the second scenario, only one parameter is changed at a time. According to the values presented in Tables 8.7, 8.8, and 8.9, the attacked parameter has a much larger standard deviation than the normal vehicles. This increase is due to the random values given to the selected parameter (speed, acceleration and heading) and thus having a larger range than the normal messages. The frequency of the generated messages is the same for normal and attacking vehicles. So the number of attack messages is much smaller than for the DoS. In this scenario, the number of attackers varies from 4 to 20% and the number of attacking messages from 2.37 to 14.01%. The file size of the created datasets is also smaller ranging from 4.7 to 511 MB.

Due to the characteristics of the attacks, the DoS attack produces much more messages. The non-attack and DoS messages are more than 97% of the training and test datasets' total data, as shown in Table 8.10.

Table 8.10: Contents of the test and training datasets per message type

Parameter	Training		Test	
	Value	% of Total	Value	% of Total
Messages (Total)	2,491,271	100.00	17,237,722	100.00
Non-Attack	1,508,873	60.57	9,062,023	52.57
DoS	912,875	36.64	7,756,817	45.00
Fab. Speed	30,002	1.20	172,892	1.00
Fab. Acc	23,819	0.95	62,686	0.36
Fab. Heading	15,702	0.63	183,304	1.06

### 8.3.2 Datasets Evaluation in the Context of an Intrusion Detection System

The datasets created in Chapter 5 are composed of data collected from simulations performed in multiple geographical maps. In this Section these are evaluated to gauge their usability for use in further works. The evaluation is made in the context of an IDS and is divided into three: their usability, the advantage of having multiple datasets, and the detection's behavior depending on its network location.

The first evaluation is straightforward; the datasets are only being fed to an ML algorithm gauging the possibility of detecting any anomaly. Then, the second evaluation will use the multiple datasets and compare

the results of having one, two, three, four, or five datasets to train the IDS. Finally, the last evaluation will assume a more realistic scenario where the IDS will not behave as an oracle but will only listen to messages received by its nodes. In this case, the goal is to simulate a cluster-based IDS implementation, such as in Chapter 6, where the IDS is divided into multiple clusters located in different network places.

All evaluations will use the same ML algorithm. The goal is only to vary the data fed to the algorithm and thus to compare the datasets.

Most of the studies use the same datasets to train and test the ML algorithms by applying some division on the dataset. However, if the data is not diverse enough, the train and test data will be very similar. It overfits the model resulting in outstanding results in the test data but bad performance in any other datasets, as shown in Section 8.2.

The tests were made using an application developed in **Python**, using the **sklearn** library [111]. The algorithm used was NN configured with the parameters indicated in Table 8.11. It uses 3 layers, with 2, 4, and 2 perceptrons and a toll of 0.0001. This value indicates that if the loss or score is not improving by at least 0.0001 during 5 consecutive "number iteration no change", the learning rate decreases. The learning rate controls how quickly or slowly the algorithm will converge. The algorithm will run for 20 epochs and use 32 (batch size) samples from the training set to calculate the error gradient. An epoch is a term that represents the number of times the ML algorithm completely analyzes the entire training dataset.

Table 8.11: ML Algorithm Configuration Parameters for Evaluation in the Context of an IDS

Parameter	Value
Algorithm	Neural Networks
Layers	2 4 2
Tol	0.0001
Learning rate init	0.1
Number iteration no change	5
Epochs	20
Batch size	32

The evaluations will be made using 5 different metrics, accuracy, precision, positive precision, negative precision, positive recall, and negative recall. Usually, only attack detection is used for the evaluation. However, the datasets used in this work are unbalanced; as it can be seen in Table 8.12, the attack messages represent most of the datasets. Thus, if all were correctly classified, the model would have a false good performance.

The data used consists of 7 different datasets that originated from 7 different geographical maps. These are marked from 1 to 7. In all the tests, the dataset originating from the geographical Map 7 will be used as the test dataset. It has a larger number of messages, vehicles, and more types of roads. The others will serve to train the model.



Table 8.12: Dataset Contents - Singular maps

Dataset	Normal (%)	Attack (%)	Total
Map 1	23193 (35%)	42264 (65%)	65457
Map 2	100707 (44%)	127946 (56%)	228653
Map 3	55621 (24%)	178999 (74%)	234620
Map 4	305170 (27%)	815990 (73%)	1120160
Map 5	79893 (35%)	150593 (65%)	230432
Map 6	83000 (20%)	330231 (80%)	413231
Map 7	1555401 (17%)	7756817 (83%)	9312218

The tests are divided into 3 as follows:

**Evaluating as an oracle with single-source datasets** Test if it is possible to detect the attacks in the dataset using the [IDS](#) as an oracle. This test will also allow verifying if the datasets are diverse enough. If all the tests have excellent accuracy, it means that probably the datasets are very similar between them and with the test dataset.

**Evaluating as an oracle with multiple sources datasets** Test if there is any advantage in having various datasets from various sources. In this test, the various datasets that originate from different geographical maps will be grouped together, enabling to gauge the advantage of having multiple sources.

**Evaluating using from different cluster levels** Evaluate how the [ML](#) behaves depending on the network location, using the cluster-based architecture from Chapter 6 as a blueprint for the cluster locations. In this case, the [IDS](#) will not act as an oracle but will only have access to the messages received by its nodes.

### Evaluating as an Oracle With Single Source Datasets

This type of [IDS](#) is the most commonly found in the literature. It has the ability to listen to all the messages in the network.

This evaluation aims to gauge how well the [IDS](#) would perform if it only had access to the data originating from one geographical Map. The test is two-fold as it will also allow verifying the diversity in the data collected. The contents of the datasets used in this test are shown in Table 8.12. This Table contains the number and percentage of normal and attack messages and the total number of messages in the dataset.

Figure 8.1 shows the results obtained from the first test. In the chart, the X-Axis represents the Map used to train the model and the Y-Axis, the accuracy (blue), negative and positive precision (orange and yellow),

and positive and negative recall (gray and blue).

The recall across all the tests was fairly equal, but only the model trained using the datasets from Map 4 and 6 performed acceptably for all the other parameters. So, if datasets obtained from Map 4 or 6 were used, the IDS would detect almost all the attacks. However, in any other situation, it would have a very low accuracy rendering the IDS useless. But, as is confirmed by Figure 8.2, the dataset's quality does not seem to depend only on its size. In this Figure, there are two scales; the left one refers to the number of attackers, normal vehicles, and total vehicles; the right one, in green, refers to the number of messages in the dataset. So, it is possible to see, that for example, that the dataset originating from Map 2 has more messages than the one obtained from Map 6 but has a much lower quality. The number of vehicles (attackers and normal) is very similar for Maps 1, 2, 5, and 6. So, it does not seem to make any difference.

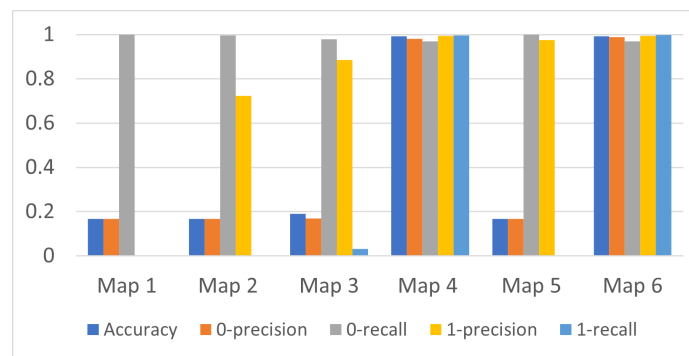


Figure 8.1: Evaluation of the Datasets Produced in each Geographical Map

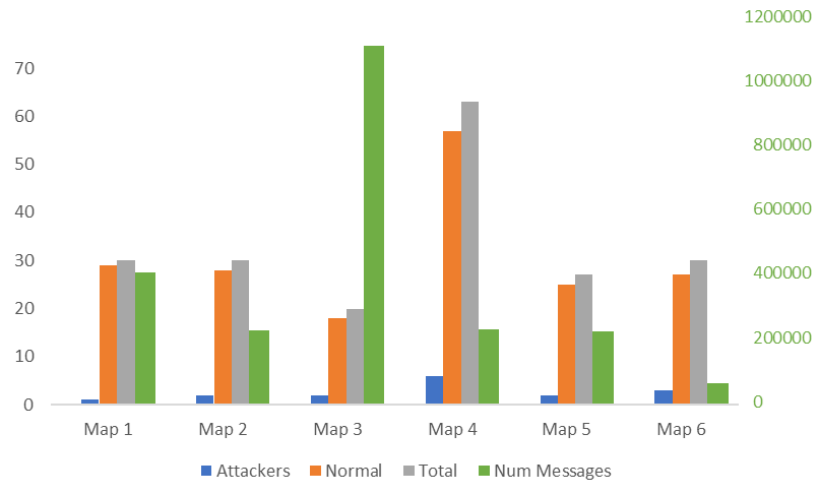


Figure 8.2: Number of Attackers (Left-Hand Side) and Messages (Right-Hand Side) per Dataset

On the plus side, this evaluation seems to indicate that the datasets are different enough between themselves and from the test dataset. Thus, the different results across them all.

## Evaluating as an Oracle with Multiple Sources Datasets

This is a more complex test that evaluates if having datasets originating from multiple sources is advantageous. To do so, the datasets were grouped into multiple test datasets using different group sizes. The datasets were joined into groups of 2, 4, 5, and 6, composing four different test datasets.

Ten groups of 2, 3 and 4 datasets were made to eliminate possible bias by combining them in multiple random ways and calculating the standard deviation of the results. The groupings containing 5 datasets only had 6 different permutations, and the one with 6 only 1 due to the number of existing datasets.

Figure 8.3 presents the results for the tests made with 2, 3, 4, and 5 datasets. To better understand the obtained results, whisker box charts were used. These allow displaying the data distribution through their quartiles. The X-axis indicates how many datasets that particular grouping has, and the Y-axis is the value for the accuracy, precision, positive recall, negative recall, negative precision, and positive precision. In this case, the name "positive" refers to a message containing an attack and "negative" for the normal messages.

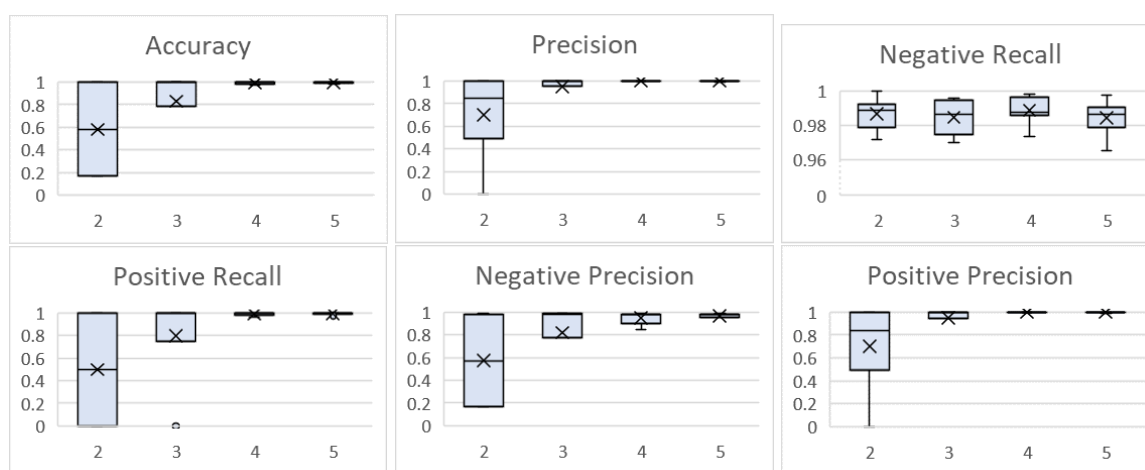


Figure 8.3: Evaluation Using 2, 3, 4 and 5 Datasets Grouping - Random Grouping

As previously mentioned, there is an asymmetry in the number of attacks and normal messages (Table 8.2). So, the precision and positive precision results are very similar, which means that the classification of the normal messages has little effect on the overall precision. Thus, the results for the total, negative and positive precision have been separated to better understand the behavior of the IDS.

The primary outcome obtained from this test is the variance of the results obtained. So, each time the number of datasets in the grouping is increased, the variance in the results decreases. This is indicated by the increasingly smaller blue square in each one of the charts. The other outcome is the average value for each parameter. As it is possible to see, it is closer to 1 with each added dataset. The only exception is the negative recall that does not change much across all parameters. Thus, with more data sources, it is possible to have better results.

Table 8.13 presents the individual results for each of the tests done for the two datasets grouping to better understand the results obtained. The first column indicates the datasets used to test the ML algorithm. The rest of the columns indicate the results obtained in the prediction: accuracy, the precision of the negative prediction, recall of the negative precision, precision of the positive prediction, and recall of the positive prediction.

Table 8.13: Dataset Evaluation - Two Dataset Grouping

Datasets	Acc	Neg Prec	Neg recall	Pos Prec	Pos Recall
2,5	0.17	0.17	1.00	0.00	0.00
5,6	0.99	0.98	0.99	1.00	1.00
1,6	0.99	0.99	0.98	1.00	1.00
1,5	0.17	0.17	1.00	0.00	0.00
2,3	1.17	0.17	0.99	0.69	0.00
4,6	0.99	0.98	0.97	0.99	1.00
1,3	0.17	0.17	0.99	0.65	0.01
5,6	0.99	0.98	0.99	1.00	1.00
3,4	0.99	0.99	0.97	0.99	1.00
2,4	0.99	0.98	0.98	1.00	1.00

Table 8.13 indicates that the variation in the detection capabilities depends on the quality of the datasets which were fed to the ML algorithm. As shown in Figure 8.1, the datasets that give better results are obtained from maps 4 and 6. Thus, if the datasets fed to the ML algorithm contain one of these, the detection rate is automatically improved, showing the importance of the diversity of the datasets.

### Evaluating from Different Cluster Levels

Unlike the other tests, in this case, the IDS will not behave as an oracle and will only have access to the messages collected by the nodes belonging to the IDS. The number of nodes of the IDS depends on its network-level location. The clusters considered defined in Section 6.1 were used for this evaluation. So, each network level will have increasingly more nodes collecting data for the IDS training. In the first level,  $L_0$ , the IDS, will only access the messages collected by that particular vehicle performing the detection. At level  $L_1$ , all the vehicles in the cluster. At the  $L_2$  level, the IDS will have access to all the messages collected by the selected vehicles in that particular geographical Map. In the last test,  $L_3$ , the IDS will have access to all the messages collected from all the selected vehicles. It is to notice that not all the vehicles in the simulation will be considered, at least for the training, only the ones composing the IDS.

The cluster definition needs to be made to divide the messages by nodes correctly. The base cluster in this particular implementation is the platooning. The Platoons were built by running several simulations and analyzing their output in conjunction with the configuration files. Then the vehicles that traveled the same

path were grouped. The simulations were performed using SUMO and VSimRTI with the code published by Gonçalves et al. [20] and available at <https://github.com/fabio-r-goncalves/dataset-collection>.

The clusters are shown in Table 8.14. The cluster-level decreases left to right, with the higher level on the left and lower on the right. The leftmost column, " $L_2$ ," identifies each of the geographical maps from which the dataset was obtained, represented by "Map x," where the x indicates the map number. Next, the column " $L_1$ " identifies each Platoon convoy that moves within each map. So, "Platoon x.y" identifies the "y" Platoon in the geographical map x. Finally, the third column,  $L_0$ , identifies the individual vehicles that compose each Platoon. The identification of each vehicle was the one used by the simulator (by convenience, it was shortened from "veh\_z" to "v\_z"). Thus, the ID of a vehicle may appear repeated in different Maps, although they refer to different vehicles.

Table 8.14: Cluster Division Using the Entities from the Dataset

$L_2$	$L_1$	$L_0$
Map 1	Platoon 1.1	v_0, v_1, v_2, v_3, v_4
	Platoon 1.2	v_14, v_15, v_17, v_21
	Platoon 1.3	v_16, v_18, v_20, v_23, v_26
Map 2	Platoon 2.1	v_0, v_6, v_12, v_18, v_24
	Platoon 2.2	v_1, v_7, v_13, v_19, v_25, v_23, v_26
	Platoon 2.3	v_32, v_35, v_37, v_40, v_43
	Platoon 2.4	v_38, v_41, v_44, v_45, v_46, v_47, v_50
	Platoon 2.5	v_49, v_52, v_54, v_56, v_59
	Platoon 2.6	v_48, v_51, v_53, v_55
Map 3	Platoon 3.1	v_1, v_3, v_7, v_10, v_12
	Platoon 3.2	v_13, v_15, v_17, v_18, v_19
Map 4	Platoon 4.1	v_4, v_5, v_7, v_10, v_11
	Platoon 4.2	v_18, v_20, v_23, v_25
	Platoon 4.3	v_32, v_25, v_27, v_40, v_43
	Platoon 4.4	v_38, v_41, v_44, v_45, v_46, v_47, v_50
	Platoon 4.5	v_49, v_52, v_54, v_56, v_59
	Platoon 4.6	v_48, v_51, v_53, v_55
Map 5	Platoon 5.1	v_2, v_3, v_5, v_7
	Platoon 5.2	v_10, v_12, v_14
	Platoon 5.3	v_19, v_21, v_23, v_25, v_26
Map 6	Platoon 6.1	v_0, v_1, v_2, v_3, v_4, v_6, v_8
	Platoon 6.2	v_14, v_15, v_17, v_19, v_21, v_24, v_27
	Platoon 6.3	v_16, v_18, v_20, v_23

Following the methodology of the previous tests, the dataset collected from Map 7 is used as the test dataset, and the ML algorithm is configured with the same parameters, as indicated in Table 8.11.

The datasets from each network location will be fed to the ML algorithm one by one. Then, the model

created will be used to evaluate the dataset from Map 7. Next, the average and standard evaluation for the accuracy, precision, negative recall, positive recall, negative precision, and positive precision for the several tests for each location will be calculated. Figure 8.4 presents the results, where X-Axis represents the cluster level that collected the data and Y the value obtained for each of the parameters.

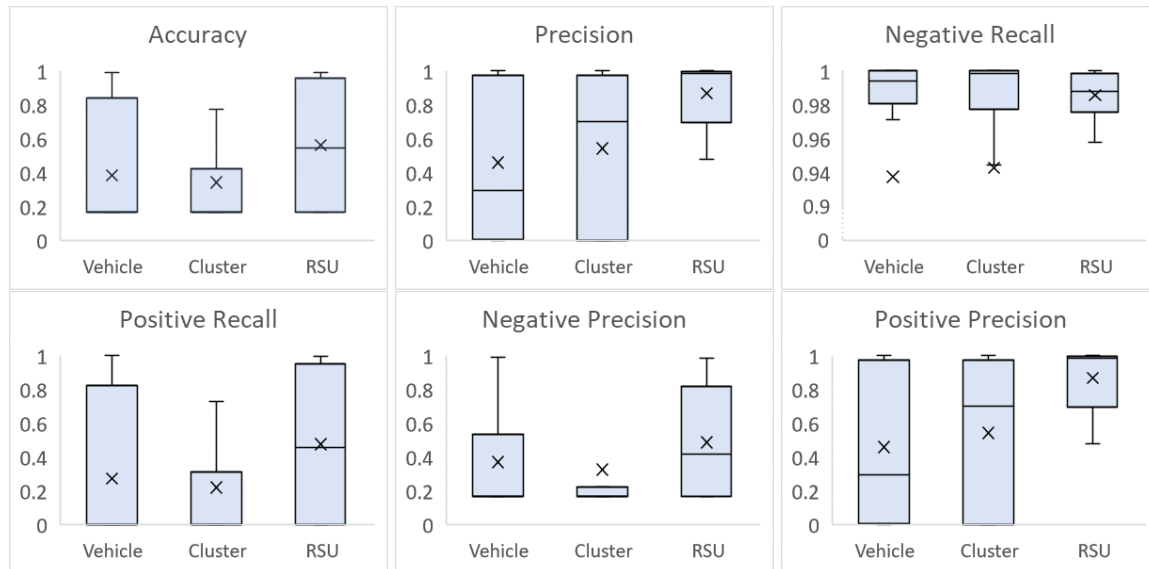


Figure 8.4: Dataset Evaluation Result per Cluster Level

Similarly to the previous test, it can be seen that the values for the precision and positive precision are very similar. This phenomenon happens due to the number of attack messages vs. the number of normal messages in the datasets (Table 8.12). Thus, it is also presented the negative precision to provide a better insight into the IDS behavior.

In this case, as there are fewer data available to train the IDS and the ML algorithm used is the same with no type of optimization, the results present a huge variance. There is a small increase in the average result for all parameters from vehicle level to RSU level, but it does not seem significant.

Due to the variance, the results presented in the charts are hard to read, so Table 8.15 presents the outcome of the various evaluations in more detail. Table 8.15 also includes the evaluation done using the data from all the RSUs. In this case, instead of using the 6 to build a training dataset, 5 were used, allowing to make various permutations of the datasets to calculate the average and standard deviation. Otherwise, only one evaluation would be possible, which would not be of much use because it would not be possible to know if the result obtained was an outlier. If one of the datasets had a much better performance than the others, it could possibly influence the results. Thus, using several permutations, it is possible to understand better the behavior of having data from multiple RSUs.

In Table 8.15, it is possible to see the average and standard deviation for the following parameters: accuracy, precision, the precision of the negative results, recall of the negative prediction, precision of the

Table 8.15: Dataset Evaluation Result per Cluster Level

Data	Acc.	Prec.	N. Prec	N. Recall	P. Prec	P. Recall
Vehicle	$0.38 \pm 0.37$	$0.46 \pm 0.42$	$0.37 \pm 0.34$	$0.94 \pm 0.20$	$0.46 \pm 0.42$	$0.27 \pm 0.43$
Cluster	$0.34 \pm 0.31$	$0.54 \pm 0.41$	$0.32 \pm 0.31$	$0.94 \pm 0.19$	$0.54 \pm 0.41$	$0.22 \pm 0.39$
RSU	$0.62 \pm 0.39$	$0.88 \pm 0.19$	$0.55 \pm 0.35$	$0.99 \pm 0.01$	$0.89 \pm 0.19$	$0.55 \pm 0.47$
All	$0.84 \pm 0.30$	$0.98 \pm 0.00$	$0.76 \pm 0.30$	$1.00 \pm 0.00$	$1.00 \pm 0.00$	$0.81 \pm 0.36$

positive prediction, and recall of the positive prediction. In the Table, it can be seen more clearly the variance of the results. However, although the results obtained for the first two levels (vehicle and cluster) these improve for the RSU and all the RSUs. Thus the higher the node is located on the network, the more data it has more data available (Table 8.16) and more diverse, providing datasets for training with more quality, resulting in better detections.

Table 8.16: Number of Messages per Cluster Level

Level	Min	Max	Average
Vehicle	788	38480	11095
Cluster	4165	157398	56149
RSU	56537	472849	215238
All	-	1082138	-

## 8.4 Hierarchical IDS Results Evaluation

ML uses data mining techniques to infer knowledge from collected data [55] and can be used to find patterns in already gathered data. ML algorithms can be classified into supervised, unsupervised, and semi-supervised [56]. This work focus on supervised learning, which assumes that each instance has a correspondent label, meaning that all trained data needs to be labeled. There is a vast variety of these types of algorithms, for example, Decision Trees [58], NN [60], SVM [60], etc.

The ML algorithms are trained using the datasets indicated in Section 5. The contents of the training and test datasets are presented in Table 8.10.

Due to the simplicity of testing, as it needs no coding to test several algorithms easily, Weka [112] was used. Weka is a well-known tool in Java that has several ready-to-use algorithms. The test was performed by first building a model using the training data; then, the model was used to classify the test datasets.

### 8.4.1 Performance Evaluation

Table 8.17 presents the results obtained from the classification of the test datasets using multiple algorithms. The first three columns represent the accuracy, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values. The individual TPR and FPR are then presented for each different attack. Finally, the last two columns show the average TPR and FPR for each algorithm (the TPR and FPR for the normal messages are not included in the calculated average).

Usually, the first three columns are representative of the model quality. However, this is not accurate in this case due to the characteristics of the datasets. Table 8.10 shows an unbalance between the collected datasets as the nonattack and DoS messages make 97% of the total messages. So, if the ML algorithm can only detect both of these correctly, it will have an accuracy of 97%.

Thus, instead of looking at the accuracy, the individual TPR and FPR can help find a better solution. The goal is to find an algorithm that maximizes the TPR while minimizing the FPR. The average TPR in the results presented varies from 0.26 to 0.88, with both J48 and RF performing the best. For the average FPR, all algorithms perform quite well, with very low values.

The results from Table 8.17 indicate that MLP can better detect the normal messages and DoS attacks, with 1.00 TPR for both and only 0.05 FPR for the normal messages. Nonetheless, it cannot detect any other attack accurately. RF can better detect Speed Fabrication attack and Heading Fabrication attack with a correspondent 0.78 and 0.90 TPR and 0.01 and 0 FPR. The Acceleration Fabrication attack is better detected using J48, which can do so with 0.78 TPR and 0.01 FPR. However, the best overall performance indicated by the average TPR and FPR shows a tie between RF and J48 (0.88 average TPR) with a slight advantage for the RF that presents a higher accuracy.

Table 8.17: Evaluation Results of the ML algorithms

Algorithm				Normal		DoS		Speed		Acc		Heading		Avg TPR	Avg FPR
	Accuracy	MAE	RMSE	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR		
Random Forest	<b>0.98</b>	0.02	<b>0.08</b>	0.98	<b>0.02</b>	0.98	<b>0.00</b>	<b>0.77</b>	0.01	0.77	<b>0.00</b>	<b>0.90</b>	<b>0.00</b>	<b>0.88</b>	<b>0.01</b>
MLP	<b>0.98</b>	<b>0.01</b>	0.09	<b>1.00</b>	0.05	<b>1.00</b>	<b>0.00</b>	0.26	0.00	0.10	<b>0.00</b>	0.00	<b>0.00</b>	0.47	0.06
J48	0.97	<b>0.01</b>	0.11	0.97	<b>0.02</b>	0.98	<b>0.00</b>	0.77	0.01	<b>0.78</b>	0.01	0.89	0.01	<b>0.88</b>	<b>0.01</b>
REP Tree	0.97	<b>0.01</b>	0.10	0.97	<b>0.02</b>	0.99	<b>0.00</b>	0.71	0.01	0.62	0.01	0.87	<b>0.00</b>	0.83	<b>0.01</b>
LMT	0.97	<b>0.01</b>	0.10	0.98	0.03	0.98	<b>0.00</b>	0.76	<b>0.00</b>	0.76	<b>0.00</b>	0.89	<b>0.00</b>	0.87	<b>0.01</b>
Random Tree	0.97	<b>0.01</b>	0.11	0.97	0.03	0.98	0.01	0.69	0.01	0.61	0.01	0.86	<b>0.00</b>	0.82	<b>0.01</b>
Hoeffding Tree	0.96	0.05	0.12	0.98	0.06	0.97	<b>0.00</b>	0.64	<b>0.00</b>	0.28	<b>0.00</b>	0.62	<b>0.00</b>	0.70	<b>0.01</b>
Logistic	0.95	0.04	0.13	0.99	0.09	0.96	0.01	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.39	0.02
OneR	0.94	0.03	0.16	0.99	0.13	0.92	0.01	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.38	0.03
Decision Stump	0.94	0.04	0.16	0.99	0.13	0.92	0.01	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.38	0.02
SMO	0.92	0.24	0.32	0.91	0.05	<b>1.00</b>	0.10	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.38	0.02
PART	0.97	0.01	0.11	0.97	0.02	0.99	<b>0.00</b>	0.66	0.00	0.63	0.01	0.84	<b>0.00</b>	0.82	<b>0.01</b>
Naive Bayes	0.85	0.06	0.24	0.74	0.03	0.99	0.21	0.75	0.01	0.53	0.01	0.53	0.01	0.65	0.06
Decision Table	0.66	0.18	0.29	<b>1.00</b>	0.71	0.31	<b>0.00</b>	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.00	<b>0.00</b>	0.26	0.14

Table 8.18 presents the sizes of the models created by Weka for each algorithm and the times needed for training and testing it. This table can help to decide which algorithm is better suited for each level, where



the lower layers benefit from smaller models (less traffic) and quicker detection. The results presented show that Logistic Model Tree (LMT) is clearly the slowest to train. However, RF is the slowest to evaluate all the messages from the test dataset. DS, while not being the quickest to train it, is the quickest to detect the attacks. Moreover, it is the lighter model needing only 3 KB to be transmitted. Additionally, the DS algorithm can easily be translated into a few if statements, needing, in reality, only a few bytes. On the other side, RF is the heavier algorithm needing 54,497 KBs.

Table 8.18: Comparison of the Size and Elapsed Time Taken During Training and Testing

Algorithm	Train Time (s)	Test Time (s)	Size (KB)
LMT	21879	23	1225
Decision Table	4514	169	29684
PART	3128	281	4860
Random Forest	3014	476	54497
SMO	2688	23	10
MLP	1176	25	16
J48	335	31	1024
Logistic	139	30	9
REPTree	123	19	791
HoeffdingTree	10	77	512
Decision Stump	7	<b>15</b>	<b>3</b>
OneR	7	29	17
Naive Bayes	<b>4</b>	79	5

### 8.4.2 Ensemble based Evaluation

The results do not need to be applied individually. Using an ML technique called ensemble learning, multiple algorithms can be used together to take advantage of each algorithm's properties. This can be especially useful in this case, as each algorithm performs well in one attack but poorly in others. Weka supports two ensemble techniques that have been used in this work; stacking and voting. Both methods combine multiple algorithms to achieve better results. Stacking calculates each model's outputs and then applies another ML algorithm (meta classifier) to the output. Voting may use different methods: the most voted option, average of probabilities, minimum of probabilities, maximum probability, and product of probabilities.

Both ensemble techniques were used to evaluate the datasets using the following algorithms (the better performing for each parameter in the previous evaluation): RF, J48, and MLP. The configurations of the selected algorithms are presented in Table 8.19. The algorithm used as the meta classifier was RF. It was the algorithm with the better overall performance, being the ideal candidate.

Additionally, a custom stacking technique was implemented. Instead of using the meta classifier, it

Table 8.19: Configuration of the Algorithms for the Ensemble Learning

<b>MLP</b>		<b>RF</b>		<b>J48</b>	
batchSize	100	batchSize	100	batchSize	100
numDecimalPlaces	2	numDecimalPlaces	2	numDecimalPlaces	2
hiddenLayers	a	bagSizePercent	100	confidenceFactor	0.25
learningRate	0.3	maxDepth	0	minNumObj	2
momentum	0.2	numExecutionSlots	1	numFolds	3
seed	0	numFeatures	0	seed	1
trainingTime	500	numIterations	100		
validationSetSize	0	seed	1		
validationThreshold	20				

uses the algorithms in the order that can benefit each one's properties. So, each instance of the dataset is evaluated using the following algorithm: (1) Apply [RF](#). If speed or heading fabrication attack is detected, the algorithm stops; otherwise, it goes to the next step. [RF](#) can detect Speed and Heading fabrication with almost no False-Positives, so if one of those attacks is detected, it has a high probability of being correct. (2) Apply [J48](#), if an acceleration fabrication attack is detected, the algorithm stops; otherwise, it goes to the next step. The reasoning is similar to the previous step. (3) Apply [MLP](#), this is the last step, and if no attack is detected at this point, then the message is considered normal. [MLP](#) is the last algorithm to be applied as it is the one with a higher [FPR](#) at detecting normal messages. Otherwise, it could classify the message as normal, even if it was an attack. The algorithm is shown in [Figure 8.5](#).

The results from the ensemble learning are presented in [Table 8.20](#). These show that the detection capabilities can be increased using an ensemble technique, mainly the custom one. It has a small increase over the best performant single algorithm, [RF](#), ranging from 0.01 to 0.02 across all the message types, with the overall average [TPR](#) also increasing from 0.88 to 0.89.

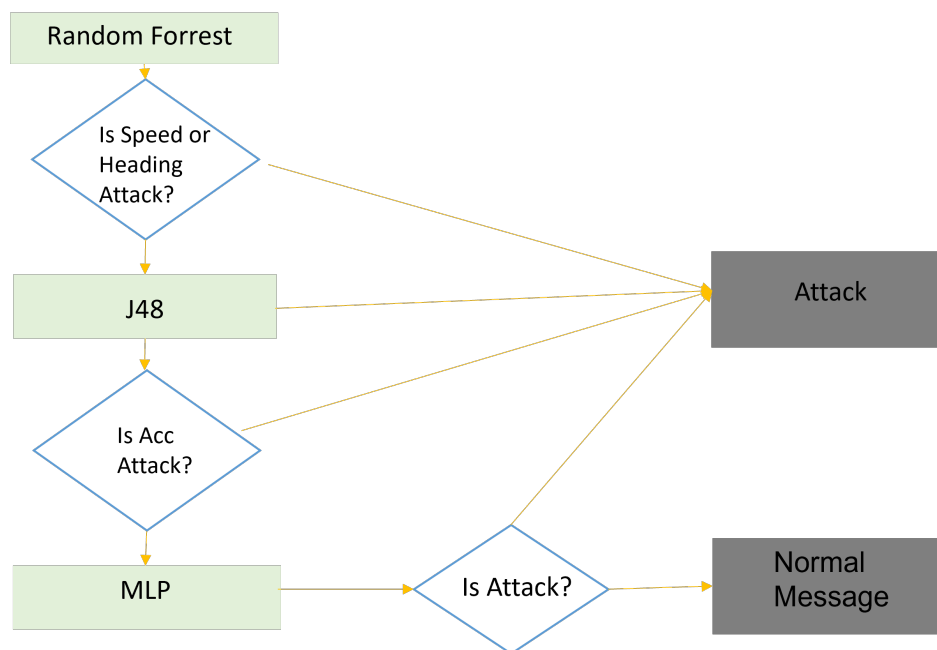


Figure 8.5: Custom Stacking Algorithm.

Table 8.20: Classification Results Using Ensemble Learning

Ensemble	Normal			DoS		Speed		Acc		Heading		Avg TPR	Avg FPR
	Accuracy	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR		
Stacking Custom	<b>0.98</b>	0.98	<b>0.01</b>	<b>0.99</b>	<b>0.00</b>	<b>0.79</b>	<b>0.00</b>	<b>0.79</b>	0.01	<b>0.90</b>	<b>0.00</b>	<b>0.89</b>	<b>0.00</b>
Stacking	<b>0.98</b>	0.98	0.02	0.98	<b>0.00</b>	<b>0.79</b>	<b>0.00</b>	0.67	<b>0.00</b>	<b>0.90</b>	<b>0.00</b>	0.86	<b>0.00</b>
Vote Major	<b>0.98</b>	<b>0.99</b>	0.02	<b>0.99</b>	<b>0.00</b>	0.76	<b>0.00</b>	0.72	<b>0.00</b>	0.87	<b>0.00</b>	0.87	<b>0.00</b>
Vote Average	<b>0.98</b>	<b>0.99</b>	0.02	<b>0.99</b>	<b>0.00</b>	0.75	<b>0.00</b>	0.71	0.00	0.88	<b>0.00</b>	0.86	<b>0.00</b>
Vote Maximum	0.97	0.97	0.03	0.98	<b>0.00</b>	0.77	<b>0.00</b>	0.77	0.01	0.61	<b>0.00</b>	0.82	0.01
Vote Product	0.97	0.97	0.03	0.98	<b>0.00</b>	0.77	<b>0.00</b>	0.76	0.01	0.84	<b>0.00</b>	0.86	0.01
Vote Minimum	0.97	0.97	0.03	0.98	<b>0.00</b>	0.77	<b>0.00</b>	0.77	0.01	0.61	<b>0.00</b>	0.82	0.01

### 8.4.3 Rule-based Evaluation

Despite the accuracy of the algorithms, the lower levels need to have quick and light detections. DS is a one-level decision tree. It creates very basic rules, usually an if statement that can quickly perform decisions. Additionally, it has a smaller size and can easily be sent through the network. So, a DS algorithm was fed datasets containing only one type of attack at a time. As it is a one-level decision tree, it will only create a rule for each time it is trained. Therefore, in reality, four different rules are being created. The test dataset used was the same as in the other tests. The results obtained are indicated in Table 8.21. Each line corresponds to a different attack. The columns represent the TPR and FPR for the detection of normal messages or attacks. As the DS algorithm is a one-step decision tree, it will only create a rule for the attack fed. Thus, it will only detect the attack it was created to detect. It can detect Normal messages and DoS quite well (0.99 TPR), but it is quite poor in the other attacks. On the plus side, it has very low FPR when detecting any of the attacks. Thus, it has a very low danger of discarding authentic messages.

In all the analyzed algorithms, detecting the fabrication attacks seems less accurate than detecting DoS. The unbalance in detecting the fabrication attacks may be due to the number of messages for each attack class. DoS has more messages, allowing the ML algorithm to be better fitted. The same happens for the DS. As it is shown, the attack with more messages, DoS, has better detection performance.

Table 8.21: In-depth Evaluation of the Decision Stump Algorithm

Attack Type	Normal		Attack	
	TPR	FPR	TPR	FPR
DoS	0.99	0.10	0.94	0.01
Speed	1.00	1.00	0.47	0.00
Acceleration	1.00	1.00	0.00	0.00
Heading	0.99	0.99	0.47	0.00

# Chapter 9

## Conclusions and Future Work

This final Chapter describes a brief and critical analysis of the obtained results and the conclusion taken from the overall research work. Additionally, the limitations presented by the research are described, indicating possible biased results and discussing other approaches. Finally, are presented the future steps, indicating future research work that may complement the work presented in this thesis.

### 9.1 Conclusions

In this thesis, a new Intrusion Detection System (IDS) design targeted at vehicle networks was proposed. It is based on a hierarchical architecture divided into four different levels that attribute different roles and functionalities to each cluster's nodes, allowing each node to perform the role and use the detection algorithm that better suits it. The nodes located in the upper nodes of the architecture are usually more powerful (infrastructural entities) that can use more complex detections. The lower nodes are the opposite. They have much lower capabilities but need quick decisions.

The first step of this research work was a Systematic Literature Review (SLR), which is one of the most widely accepted ways to survey the literature. The SLR defines a methodology to survey multiple databases that must be carefully followed. Thus, providing results in a more systematic way that can be easily replicated and verified. The goal of the SLR was to survey the existing solutions consisting of Machine Learning (ML)-based IDSs for Vehicular Ad hoc Networks (VANETs), providing insight into frequently used ML algorithms, simulation frameworks, tools, and datasets. The results showed that most works use one of the popular, Network Simulator 2 (ns-2) or Network Simulator 3 (ns-3), network simulator versions, Simulation of Urban Mobility (SUMO) as the traffic simulator and Neural Networks (NN) as ML algorithm. However, most works failed to present detailed information about the used datasets nor made them available. Consequently, the corresponding published results are very hard to verify.

The architecture of the IDS implies multiple communications between the multiple entities, either inter or intra-level. So, a security framework for VANETs was surveyed. This framework should fulfill the VANET requirements and facilitate the multiple communication types needed by a heterogeneous network like VANET. Due to the characteristics of this type of network, there is a considerable volume of research work on this area. However, the found solutions present some drawbacks. So, it was decided to design a security model. The designed model, Vehicular Ad hoc Network Public Key Infrastructure and Attribute-Based Encryption with Identity Manager Hybrid (VPKIbrID), uses a multitude of technologies to provide a hybrid solution that can tackle the individual drawbacks. It provides two encryption types, VPKIbrID Public Key Infrastructure (VPKIbrID-PKI) and VPKIbrID Attribute Base Encryption (VPKIbrID-ABE), facilitating both unicast and multicast/broadcast communications. The first encryption type, VPKIbrID-PKI, uses the most common Public Key Infrastructure (PKI) encryption to provide a more lightweight encryption but only unicast communications. The second, VPKIbrID-ABE, uses Attribute-Based Encryption (ABE). It is a less utilized encryption mechanism but provides encryption for multiple targets, optimal for multicast/broadcast.

The designed IDS architecture is based on a hierarchy, grouping entities with similar capabilities and characteristics, allowing the attribution of the optimal detection techniques and roles to each node. The architecture is divided into 4 different levels, each having multiple clusters. The architecture's levels are the following:  $L_0$ , a single-vehicle;  $L_1$ , a cluster of vehicles;  $L_2$ , a cluster of Road Side Units (RSUs);  $L_3$  backend server with high Central Processing Unit (CPU) capacity.

Usually, to build the clusters, a clustering algorithm is needed. In this work, to facilitate the clustering process, the cluster in level  $L_1$  was built using an already existing application that groups vehicles that travel together in the same path at the same speeds, platooning. This application already provides all the tools needed for the vehicles grouping and leader selection. For level  $L_2$ , the RSUs were grouped using the natural geographical division from the collected datasets. So, at this level, the IDS will have access to all the messages from a particular geographical area. The levels  $L_0$  and  $L_3$  are already defined, being the first a vehicle on the road, and latter is any specific - edge or cloud based - sink node where all the messages received by lower hierarchy nodes are collected and subsequently processed.

So, to accomodate, validate and test this new paradigm, the security model designed was adapted for the usage in the platooning, allowing the platooning vehicles to use this framework to communicate securely, taking advantage of its properties.

Due to the lack of available datasets in the literature, as found in the performed SLR, a methodology to produce VANET messages and collect datasets was designed. It allowed to generate multiple datasets originating from 7 different geographical maps, increasing the randomness introduced and decreasing the probability of creating biased detection. These synthesised datasets contain vehicular messages, both normal and attacks. The attacks produced are Denial of Service (DoS) and multiple fabrication attacks. In the DoS, the attacker vehicle produces messages similar to the other vehicles but at a much higher frequency. The

fabrication attack simulates a faulty sensor or an attacker that tries to convey false information about the ego-vehicle function parameters (such as velocity, acceleration) of the car. In this case, the following parameters are targeted: speed, acceleration, and heading. As one dataset is generated for each attack in each geographical map, a total of 42 different datasets were generated. These datasets were completely disclosed, published and made available at a public data repository for third-party evaluation and to facilitate future research works.

The produced datasets have been evaluated in three different situations, each allowing a particular conclusion. These were the following: evaluate the datasets as an oracle with single-source datasets, evaluate the datasets as an oracle with multiple source datasets, and evaluate the datasets from different cluster levels. The first evaluation allowed to conclude that it is possible to detect attacks from the collected datasets. However, the datasets seem different enough as the detection accuracy varied depending on the source used. The second test showed that the detection could benefit from having datasets from multiple different sources. Moreover, using multiple datasets allow compensating data from poor datasets during the training phase, with datasets with more quality, decreasing the probability of having overfitted models. The last test served to understand which levels of the architecture would be able to detect attacks according to the data received from their nodes. This test showed that the upper level can have a more accurate detection and would probably be the best place to use the received data to train the ML models.

Initial tests were made without any optimization of the algorithms, as the goal was only to understand the behavior of the IDS depending on the data received. So, to better understand which ML algorithm should be used in each layer, multiple ML algorithms were used to analyze the datasets. The results showed that the best overall performant algorithm is Random Forest (RF), closely followed by J48. However, for each parameter individually, the best performant algorithms are Multilayer Perceptron (MLP) to detect normal and DoS attacks, J48 to detect Acceleration Fabrication attacks, and RF to detect Speed and Heading Fabrication attacks. Nonetheless, accuracy is not the only important metric. Depending on the level, the size and performance can also be important. The slowest algorithm to train is Logistic Model Tree (LMT), RF is the slowest to analyze the dataset and also produces the biggest model, Decision Stump (DS) is the quickest to perform decisions and produces the smaller model. Additionally, due to DS construction, it can be quickly translated into a few if statements.

Moreover, instead of using a single algorithm to perform detection, these can be grouped into an ensemble algorithm, thus using the characteristics of the multiple algorithms to have a more accurate decision. So, the best performant algorithms from the previous evaluation (MLP, J48, and RF) were used to build an ensemble algorithm. The best performance was obtained using a custom ensemble algorithm, with better results than each of the individual algorithms.

However, the algorithms with better accuracy may not be the best suited for all the levels. At the lowest level, perhaps the best solution is to use an algorithm that can perform quick detections. So, using the lighter algorithm, DS, more specific tests were made, concluding that it is useful in detecting DoS attacks although

not as accurate as other more complex algorithms.

Using the security model designed and the results from the evaluation of the ML algorithms, the final architecture design was made. It uses the VPKIbrID model to secure all the communications between the node. Due to the nature of the interactions, the most used model is the VPKIbrID-ABE, which can be particularly useful when taking advantage of its key caching mechanism. The  $L_0$  level uses DS, although only being able to detect DoS accurately, it is a good first level of defense, performing very quick detections and being very light, allowing an easy transmission to all the nodes. The level  $L_1$  is composed of the same node types as the previous the detection at this level does not present any advantage. The nodes at  $L_1$  will only be responsible for forwarding the received messages to the upper nodes. The level  $L_2$  uses RF to perform detection. These are infrastructural entities with more processing power, being able to use a more complex and slower algorithm. Finally, to the level  $L_3$  are reserved the most complex and CPU-heavy operations. So, this level will perform detection using the ensemble algorithm, the most complex algorithm. Additionally, as it has a wider view of the network and access to more diverse data, it will use the received data to build rules and models to the other levels.

If any level (except  $L_0$ ) detects any attack, a system-wide response should be triggered, warning the other vehicles, putting the attacker on a blacklist, and, if needed, warning the authorities. As the first level uses a less accurate algorithm, it should only discard the attack messages, letting the decision off triggering a system-wide warning to the upper levels.

## 9.2 Limitation and Future Work

The presented work comprises of an ML-based IDS, which heavily relies on the usage of datasets — these present one of the most prominent limitations at three different levels, data origin, attacks, and size.

The used datasets were self-produced and synthesized using simulation. However, this could introduce bias in the results. The datasets should have been produced by third parties and, ideally, obtained from the real world. On the plus side, the dataset synthesized were made available to the public, along with the methodology to produce them, allowing third parties to evaluate them and enable the scrutiny of the results obtained. Additionally, this thesis work is aligned with an R&D project with the collaboration with Boshc Car Service, and it is expected to obtain real-world communication data in the near future.

As for the attacks, the used datasets contained only two different attacks, DoS and Fabrication. Although useful to test the methodology designed, it limits the detection capabilities of the dataset. Moreover, some attacks may be undetected by the IDS. Nonetheless, the implementation used to synthesize the datasets can be easily extended to produce new attacks. Additionally, the application was published on a public platform (Github), enabling the collaboration of other researchers in the implementation of new functionalities.



Although the results obtained in this work are satisfactory, the evaluation of the datasets using different clusters produced a variance higher than expected, difficulting the gauging of the exact behavior of the first two levels. This behavior is possibly caused by the lack of more datasets from different geographical areas. This limitation can be solved by following the same steps as for the number of attacks.

The designed architecture uses [VPKIbrID](#) as the security communications framework. The model designed is built using well-established and robust security measures and cryptographic primitives. However, despite of the individual strength of each methodology, the overall model may have vulnerabilities. So, it is a goal to perform a formal verification of the security model. Moreover, most of the performance tests using the [VPKIbrID](#) security model used simple applications that generate a low quantity of messages such as beacons or sporadic messages such as the one from platooning. So, more complex tests are to be made using more demanding applications, such as streaming applications.

The novel [IDS](#) design presented shows a hierarchy-based architecture. However, most of the clustering is based on simple geographical division or applications that already group vehicles (platooning). The multiple existing clustering algorithms in the literature should be evaluated, trying to gauge the best suited for these environments.

Due to the division of the network, needing the multiple entities of the system to constantly communicate, a huge quantity of data is generated. The communication bandwidth needed to transmit this quantity of data can overload any entity and be especially complex in a volatile environment such as [VANETs](#). So, a future research point is to survey and evaluate multiple compression algorithms to decrease the size of the data needed.

# Bibliography

- [1] ETSI. ETSI TS 102 940 V1.1.1 Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management, 2012.
- [2] Maxim Raya and Jean-Pierre Hubaux. The security of VANETs. Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks - VANET '05, pages 93–94, 2005.
- [3] Ahmed Hesham, Ayman Abdel-Hamid, and Mohamad Abou El-Nasr. A dynamic key distribution protocol for PKI-based VANETs. In IFIP Wireless Days, volume 1, pages 1–3, oct 2011.
- [4] IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages. IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013), pages 1–240, mar 2016.
- [5] Lina Bariah, Dina Shehada, Ehab Salahat, and Chan Yeob Yeun. Recent advances in VANET security: A survey. 2015 IEEE 82nd Vehicular Technology Conference, VTC Fall 2015 - Proceedings, 2016.
- [6] D Huang, X Hong, and M Gerla. Situation-aware trust architecture for vehicular networks. IEEE Communications Magazine, 48(11):128–135, nov 2010.
- [7] Mohamed Nidhal Mejri, Jalel Ben-Othman, and Mohamed Hamdi. Survey on VANET security challenges and possible cryptographic solutions. Vehicular Communications, 1(2):53–66, 2014.
- [8] Haowen Tan, Ziyuan Gui, and Ilyong Chung. A Secure and Efficient Certificateless Authentication Scheme With Unsupervised Anomaly Detection in VANETs. IEEE Access, 6:74260–74276, 2018.
- [9] M Erritali and B El Ouahidi. A review and classification of various VANET Intrusion Detection Systems. In 2013 National Security Days (JNS3), pages 1–6, apr 2013.
- [10] L.O. Anyanwu, J. Keengwe, and G.a. Arome. Scalable Intrusion Detection with Recurrent Neural Networks. Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, 6(1):21–28, 2010.

- [11] Sofiane Amara, Joaquim Macedo, Fatima Bendella, and Alexandre Santos. Group formation in mobile computer supported collaborative learning contexts: A systematic literature review. *Educational Technology and Society*, 19(2):258–273, 2016.
- [12] B. Ribeiro, F. Gonçalves, A. Santos, M.J. Nicolau, B. Dias, J. Macedo, and A. Costa. Simulation and testing of a platooning management protocol implementation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10372 LNCS, 2017.
- [13] Susana Sousa, Alexandre Santos, António Costa, Bruno Dias, Bruno Ribeiro, Fábio Gonçalves, Joaquim Macedo, Maria João Nicolau, and Óscar Gama. A New Approach on Communications Architectures for Intelligent Transportation Systems. *Procedia Computer Science*, 110:320–327, 2017.
- [14] Fábio Gonçalves, Alexandre Santos, António Costa, Bruno Dias, Bruno Ribeiro, Joaquim Macedo, Maria João Nicolau Nicolau, Susana Sousa, Oscar Gama, Sara Barros, and Vadym Hapanchak. Hybrid Model for Secure Communications and Identity Management in Vehicular Ad Hoc Networks. *9th International Congress on Ultra Modern Telecommunications and Control Systems (ICUMT’2017)*, pages 414–422, 2017.
- [15] Bruno Ribeiro, Fábio Gonçalves, Vadym Hapanchak, Óscar Gama, Sara Barros, Paulo Araújo, António Costa, Maria João Nicolau, Bruno Dias, Joaquim Macedo, and Others. PlaSA-Platooning Service Architecture. In *Proceedings of the 8th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, pages 80–87, 2018.
- [16] Fábio Gonçalves, Bruno Ribeiro, Vadym Hapanchak, Sara Barros, Oscar Gama, Paulo Araújo, Maria João Nicolau, Bruno Dias, Joaquim Macedo, António Costa, and Alexandre Santos. Secure Management of Autonomous Vehicle Platooning. In *Proceedings of the 14th ACM International Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet’18*, pages 15–22, New York, NY, USA, 2018. ACM.
- [17] Bruno Dias, Alexandre Santos, Antonio Costa, Bruno Ribeiro, Fabio Goncalves, Joaquim Macedo, M João Nicolau, Oscar Gama, and Susana Sousa. Agnostic and Modular Architecture for the Development of Cooperative ITS Applications. *Journal of Communications Software and Systems*, 14(3):218–227, sep 2018.
- [18] Vadym Hapanchak, António Costa, Joaquim Macedo, Alexandre Santos, Bruno Dias, M. João Nicolau, Bruno Ribeiro, Fábio Gonçalves, Oscar Gama, and Paulo Araújo. Simulation and Testing of a Platooning Cooperative Longitudinal Controller. In *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, volume 267, pages 66–80, 2019.

- [19] Fabio Goncalves, Bruno Ribeiro, Oscar Gama, Alexandre Santos, Antonio Costa, Bruno Dias, Joaquim Macedo, and Maria Joao Nicolau. A Systematic Review on Intelligent Intrusion Detection Systems for VANETs. In 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pages 1–10, oct 2020.
- [20] Fábio Gonçalves, B. Ribeiro, Óscar Gama, João Santos, António Costa, Bruno Dias, Maria João Nicolau, Joaquim Macedo, and Alexandre Santos. Synthesizing Datasets with Security Threats for Vehicular Ad-Hoc Networks. In IEEE Globecom 2020: 2020 IEEE Global Communications Conference (GLOBECOM'2020), Taipei, Taiwan.
- [21] Fabio Goncalves, Joaquim Macedo, and Alexandre Santos. Evaluation of VANET Datasets in context of an Intrusion Detection System. 29th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2021), 2021.
- [22] Fabio Goncalves, Joaquim Macedo, and Alexandre Santos. Intelligent Hierarchical Intrusion Detection System for VANETs. In 2021 13th International Congress on Ultra Modern Communications and Control Systems and Workshops (ICUMT), 2021.
- [23] Fábio Gonçalves, Joaquim Macedo, and Alexandre Santos. An Intelligent Hierarchical Security Framework for VANETs. *Information*, 12(11), 2021.
- [24] F. Gonçalves, A. Santos, and J. Macedo. *Arquitectura de Segurança para a Prestação de Serviços de Saúde em Mobilidade* (in Portuguese). Master's thesis, Universidade do Minho, 2013.
- [25] Nicusor Vatra. Public key infrastructure for public administration in Romania. 2010 8th International Conference on Communications, COMM 2010, pages 481–484, 2010.
- [26] Luca Delgrossi and Tao Zhang. Public Key Infrastructure for Vehicle Networks. *Vehicle Safety Communications*, pages 209–236, 2012.
- [27] Panagiotis Papadimitratos, Ghita Mezzour, and Jean Pierre Hubaux. Certificate revocation list distribution in vehicular communication systems. *VANET'08 - Proceedings of the 5th ACM International Workshop on VehiculAr Inter-NETworking*, pages 86–87, 2008.
- [28] Kenneth P Laberteaux, Jason J Haas, and Yih-Chun Hu. Security Certificate Revocation List Distribution for Vanet. In *Proceedings of the Fifth ACM International Workshop on VehiculAr Inter-NETworking, VANET '08*, pages 88–89, New York, NY, USA, 2008. ACM.
- [29] Jason J. Haas, Yih Chun Hu, and Kenneth P. Laberteaux. Design and analysis of a lightweight certificate revocation mechanism for VANET. *VANET'09 - Proceedings of the 6th ACM International Workshop on VehiculAr Inter-NETworking*, (January 2009):89–98, 2009.

- [30] Michael E. Nowatkowski and Henry L. Owen. Certificate revocation list distribution in VANETs using most pieces broadcast. Conference Proceedings - IEEE SOUTHEASTCON, pages 238–241, 2010.
- [31] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. pages 1–15.
- [32] M S Ferdous and R Poet. Formalising Identity Management protocols. In 2016 14th Annual Conference on Privacy, Security and Trust (PST), pages 137–146, dec 2016.
- [33] D Hardt. RFC6749-The OAuth 2.0 Authorization Framework. Oct. 2012. URL: <https://tools.ietf.org/html/rfc6749> (visited on 04/24/2015).
- [34] Nat Sakimura, D Bradley, B de Mederiso, M Jones, and Edmund Jay. OpenID connect standard 1.0-draft 07, 2011.
- [35] Michael Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT). (7519), May 2015.
- [36] Richard Gilles Engoulou, Martine Bellaiche, Samuel Pierre, and Alejandro Quintero. VANET security surveys. Computer Communications, 44:1–13, 2014.
- [37] IEEE Trial-Use Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages. IEEE Std 1609.2-2006, pages 1–105, 2006.
- [38] NIST. Online, <https://www.nist.gov/> (last access 16/12/2021).
- [39] Theodore Willke, Patcharinee Tientrakool, and Nicholas Maxemchuk. A survey of inter-vehicle communication protocols and their applications. IEEE Communications Surveys and Tutorials, 11(2):3–20, 2009.
- [40] ETSI. EN 302 665 - V1.1.1 - Intelligent Transport Systems (ITS); Communications Architecture. 2010.
- [41] C Cseh. Architecture of the dedicated short-range communications (DSRC) protocol. In Vehicular Technology Conference, 1998. VTC 98. 48th IEEE, volume 3, pages 2095–2099 vol.3, may 1998.
- [42] ETSI. ETSI TS 102 637-3 V1.1.1 Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service. 2010.
- [43] A Mpitiopoulos, D Gavalas, C Konstantopoulos, and G Pantziou. A survey on jamming attacks and countermeasures in WSNs. IEEE Communications Surveys Tutorials, 11(4):42–56, 2009.
- [44] J Kuriakose, P S Sisodia, Amruth V, D K Shah, and S More. Comparative study of diverse zero-knowledge argument systems. In 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), pages 284–293, mar 2016.

- [45] P Cencioni and R di Pietro. VIPER: A vehicle-to-infrastructure communication privacy enforcement protocol. In 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems, pages 1–6, oct 2007.
- [46] D Singelee and B Preneel. Location verification using secure distance bounding protocols. In IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005., pages 7 pp.–840, nov 2005.
- [47] Pilar Pedro J. Fernandez, Jose Santa, Fernando Bernal, Antonio F. Skarmeta, and Antonio F Gomez-Skarmeta. Securing Vehicular IPv6 Communications. IEEE Transactions on Dependable and Secure Computing, 13(1):46–58, jan 2016.
- [48] B Bellur. Certificate Assignment Strategies for a PKI-Based Security Architecture in a Vehicular Network. In IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference, pages 1–6, nov 2008.
- [49] Qingzi Liu, Qiwu Wu, and Li Yong. A hierarchical security architecture of VANET. In Cyberspace Technology (CCT 2013), International Conference on, pages 6–10, nov 2013.
- [50] Asif Ali Wagan, Bilal Munir Mughal, Halabi Hasbullah, and Bandar Seri Iskandar. 2010 Second International Conference on Communication Software and Networks VANET Security Framework for Trusted Grouping using TPM Hardware. Communication Software and Networks, 2010. ICCSN '10. Second International Conference on, pages 309–312, feb 2010.
- [51] Matthias Gerlach and Felix Güttler. Privacy in VANETs using changing pseudonyms - Ideal and real. IEEE Vehicular Technology Conference, pages 2521–2525, 2007.
- [52] Björn Wiedersheim, Michel Sall, and Guillaume Reinhard. SeVeCom - Security and privacy in car2car ad hoc networks. 2009 9th International Conference on Intelligent Transport Systems Telecommunications, ITST 2009, pages 658–661, 2009.
- [53] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In George Robert Blakley and David Chaum, editors, Advances in Cryptology: Proceedings of CRYPTO 84, pages 47–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 1985.
- [54] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection in wireless network applications. Computer Communications, 42:1–23, 2014.
- [55] Christopher J Witten Ian H and Frank, Eibe and Hall, Mark A and Pal. Data Mining: Practical Machine Learning Tools and Techniques. Chapter 7. Morgan Kaufmann, 2016.

- [56] A A Aburomman and M Bin Ibne Reaz. Survey of learning methods in intrusion detection systems. In 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEEES), pages 362–365, nov 2016.
- [57] X Zhu, Z Ghahramani, and J Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In Proceedings of the 20th International Conference on Machine Learning, pages 912–919, 2003.
- [58] S B Kotsiantis, I D Zaharakis, and P E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, nov 2006.
- [59] Yasir Hamid, M Sugumaran, and V Balasaraswathi. IDS Using Machine Learning - Current State of Art and Future Directions. *British Journal of Applied Science and Technology*, 15(3):1–22, 2016.
- [60] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
- [61] Zoubin Ghahramani. Unsupervised Learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning: ML Summer Schools 2003*, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures, pages 72–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [62] Anil K Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
- [63] T Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, sep 1990.
- [64] Weiguo Sheng and Xiaohui Liu. A genetic k-medoids clustering algorithm. *Journal of Heuristics*, 12(6):447–466, dec 2006.
- [65] Michele Segata, Bastian Bloessl, Stefan Joerer, Christoph Sommer, Mario Gerla, Renato Lo Cigno, and Falko Dressler. Toward communication strategies for platooning: Simulative and experimental evaluation. *IEEE Transactions on Vehicular Technology*, 64(12):5411–5423, 2015.
- [66] Mani Amoozadeh, Hui Deng, Chen Nee Chuah, H Michael Zhang, and Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by VANET. *Vehicular Communications*, 2(2):110–123, 2015.
- [67] Mani Amoozadeh, Arun Raghuramu, Chen Nee Chuah, Dipak Ghosal, H. Michael Zhang, Jeff Rowe, and Karl Levitt. Security vulnerabilities of connected vehicle streams and their impact on cooperative driving. *IEEE Communications Magazine*, 53(6):126–132, 2015.

- [68] Mikael Asplund. Poster : Securing Vehicular Platoon Membership. IEEE Vehicular Networking Conference, pages 119–120, 2014.
- [69] Barbara Kitchenham and S Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Engineering, 2:1051, 2007.
- [70] Crossref. Online, <https://www.crossref.org/> (accessed on 04/01/2022).
- [71] Habanero. Online, <https://github.com/sckott/habanero> (accessed on 04/01/2022).
- [72] Mohammed Erritali and Bouabid El Ouahidi. A review and classification of various VANET Intrusion Detection Systems. 2013 National Security Days - 3eme Edition Des Journees Nationales de Securite, JNS3, pages 1–6, 2013.
- [73] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, and Koji Nakao. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security - BADGERS '11, pages 29–36, 2011.
- [74] Miriam Rummel and Miriam Rummel. Der Social Entrepreneurship-Diskurs. Eine Einführung in die Thematik. Wer sind Social Entrepreneurs in Deutschland?, (Cisda):21–38, 2011.
- [75] Sudip Misra, P. Venkata Krishna, and Kiran Isaac Abraham. A stochastic learning automata-based solution for intrusion detection in vehicular ad hoc networks. Security and Communication Networks, 4(6):666–677, jun 2011.
- [76] Daxin Tian, Yunpeng Wang, Guangquan Lu, and Guizhen Yu. A vehicular ad hoc networks intrusion detection system based on BUSNet. Proceedings of the 2010 2nd International Conference on Future Computer and Communication, ICFCC 2010, 1:V1–225–V1–229, 2010.
- [77] Neeraj Kumar and Naveen Chilamkurti. Collaborative trust aware intelligent intrusion detection in VANETs. Computers and Electrical Engineering, 40(6):1981–1996, 2014.
- [78] Xiaoyun Liu, Gongjun Yan, Danda B. Rawat, and Shugang Deng. Data mining intrusion detection in vehicular ad hoc network. IEICE Transactions on Information and Systems, E97-D(7):1719–1726, 2014.
- [79] Mohamed Nidhal Mejri and Jalel Ben-Othman. Detecting greedy behavior by linear regression and watchdog in vehicular ad hoc networks. 2014 IEEE Global Communications Conference, GLOBECOM 2014, (ii):5032–5037, 2014.



- [80] Khattab M. Ali Alheeti, Anna Gruebler, and Klaus D. McDonald-Maier. An intrusion detection system against malicious attacks on the communication network of driverless cars. 2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015, pages 916–921, 2015.
- [81] Khattab M. Ali Alheeti, Anna Gruebler, and Klaus D. McDonald-Maier. An Intrusion Detection System against Black Hole Attacks on the Communication Network of Self-Driving Cars. 2015 Sixth International Conference on Emerging Security Technologies (EST), pages 86–91, 2015.
- [82] Hichem Sedjelmaci and Sidi Mohammed Senouci. An accurate and efficient collaborative intrusion detection framework to secure vehicular networks. *Computers and Electrical Engineering*, 43:33–47, 2015.
- [83] Khattab M. Ali Alheeti and Klaus McDonald-Maier. Hybrid intrusion detection in connected self-driving vehicles. 2016 22nd International Conference on Automation and Computing, ICAC 2016: Tackling the New Challenges in Automation and Computing, pages 456–461, 2016.
- [84] Khattab Ali Alheeti, Anna Gruebler, and Klaus McDonald-Maier. Intelligent Intrusion Detection of Grey Hole and Rushing Attacks in Self-Driving Vehicular Networks. *Computers*, 5(3):16, 2016.
- [85] Omar Abdel Wahab, Azzam Mourad, Hadi Otrouk, and Jamal Bentahar. CEAP: SVM-based intelligent detection model for clustered vehicular ad hoc networks. *Expert Systems with Applications*, 50:40–54, 2016.
- [86] Khattab M. Ali Alheeti, Anna Gruebler, and Klaus McDonald-Maier. Using discriminant analysis to detect intrusions in external communication for self-driving vehicles. *Digital Communications and Networks*, 3(3):180–187, 2017.
- [87] Sparsh Sharma and Ajay Kaul. Hybrid fuzzy multi-criteria decision making based multi cluster head dolphin swarm optimized IDS for VANET. *Vehicular Communications*, 12:23–38, 2018.
- [88] Laisen Nie, Yongkang Li, and Xiangjie Kong. Spatio-Temporal Network Traffic Estimation and Anomaly Detection Based on Convolutional Neural Network in Vehicular Ad-Hoc Networks. *IEEE Access*, 6:40168–40176, 2018.
- [89] Tao Zhang and Quanyan Zhu. Distributed Privacy-Preserving Collaborative Intrusion Detection Systems for VANETs. *IEEE Transactions on Signal and Information Processing over Networks*, 4(1):148–161, 2018.
- [90] Ayoob Ayoob, Gang Su, and Gaith Al. Hierarchical Growing Neural Gas Network (HGNG)-Based Semicooperative Feature Classifier for IDS in Vehicular Ad Hoc Network (VANET). *Journal of Sensor and Actuator Networks*, 7(3):41, 2018.

- [91] Junwei Liang, Jianyong Chen, Yingying Zhu, and Richard Yu. A novel Intrusion Detection System for Vehicular Ad Hoc Networks (VANETs) based on differences of traffic flow and position. *Applied Soft Computing Journal*, 75:712–727, 2019.
- [92] Jay Rupareliya, Sunil Vitthani, and Chirag Gohel. Securing VANET by Preventing Attacker Node Using Watchdog and Bayesian Network Theory. *Procedia Computer Science*, 79:649–656, 2016.
- [93] Fuad A. Ghaleb, Faisal Saeed, Mohammad Al-Sarem, Bander Ali Saleh Al-Rimy, Wadii Boulila, A. E.M. Eljaily, Khalid Aloufi, and Mamoun Alazab. Misbehavior-aware on-demand collaborative intrusion detection system using distributed ensemble learning for VANET. *Electronics (Switzerland)*, 9(9):1–17, 2020.
- [94] Dimitrios Kosmanos, Apostolos Pappas, Leandros Maglaras, Sotiris Moschoyiannis, Francisco J. Aparicio-Navarro, Antonios Argyriou, and Helge Janicke. A novel Intrusion Detection System against spoofing attacks in connected Electric Vehicles. *Array*, 5(December 2019):100013, 2020.
- [95] Abdallah R. Gad, Ahmed A. Nashat, and Tamer M. Barkat. Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset. *IEEE Access*, pages 1–1, 2021.
- [96] N. Moustafa. TON-IOT. Online, <https://research.unsw.edu.au/projects/toniot-datasets> (accessed on 04/01/2022).
- [97] Ayoub Alsarhan, Mohammad Alauthman, Esra’a Alshdaifat, Abdel Rahman Al-Ghuwairi, and Ahmed Al-Dubai. Machine Learning-driven optimization for SVM-based intrusion detection system in vehicular ad hoc networks. *Journal of Ambient Intelligence and Humanized Computing*, (0123456789), 2021.
- [98] Gunasekaran Raja, Sudha Anbalagan, Geetha Vijayaraghavan, Sudhakar Theerthagiri, Saran Vaitanagarukav Suryanarayan, and Xin Wen Wu. SP-CIDS: Secure and Private Collaborative IDS for VANETs. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4385–4393, 2021.
- [99] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, CISDA'09*, pages 53–58. IEEE Press, 2009.
- [100] NSL KDD 99. Online, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (accessed on 10/01/2022).
- [101] DCAITI. VSimRTI. Online, <https://www.dcaiti.tu-berlin.de/research/simulation/> (accessed on 2020-11-12).
- [102] DCAITI. Eclipse Mosaic. Online, <https://www.eclipse.org/mosaic/> (accessed on 2020-12-16).

- [103] Vehicular Communications and Awareness Basic Service. Vehicular Communications ; Basic Set of Applications ; Part 2 : Specification of Cooperative. History, 1:1–22, 2010.
- [104] Fábio Gonçalves, Alexandre Santos, and Joaquim Macedo. V2X Security Threats, 2021.
- [105] Xianlei Ge, Qiang Gao, and Xunzhong Quan. A novel clustering algorithm based on mobility for VANET. International Conference on Communication Technology Proceedings, ICCT, 2019-October:473–477, 2019.
- [106] Hamayoun Shahwani, Toan Duc Bui, Jaehoon Paul Jeong, and Jitae Shin. A stable clustering algorithm based on Affinity Propagation for VANETs. International Conference on Advanced Communication Technology, ICACT, pages 501–504, 2017.
- [107] Xiaolu Cheng, Baohua Huang, and Wei Cheng. Stable clustering for VANETs on highways. Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing, SEC 2018, pages 399–403, 2018.
- [108] Java. Online, <https://www.java.com/> (accessed on 2021-07-03).
- [109] Maven. Online, <https://maven.apache.org/> (accessed on 2021-07-03).
- [110] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, March 2014.
- [111] scikit-learn. Online, <https://scikit-learn.org/> (accessed on 2020-12-10).
- [112] Eibe Frank, Mark A Hall, and Ian H Witten. The WEKA workbench. Data Mining, pages 553–571, 2017.