

# POLITECNICO DI TORINO

Master's Degree in COMPUTER ENGINEERING



Master's Degree Thesis

## Implementing secured messages for V2X communication

Supervisors

Prof. Claudio Ettore CASETTI

Prof. Riccardo SISTO

Candidate

Alessandro GIACCAGLINI

December 2024



# Summary

This thesis investigates the implementation of secure messaging protocols for V2X (vehicle-to-everything) communications, a technology that enables vehicles to communicate with other vehicles, infrastructure, pedestrians, and networks. The integration of V2X technologies offers considerable promise for improving road safety, optimising traffic flow and reducing the environmental impact of transportation. However, the open nature of these networks, such as VANETs (Vehicular Ad-Hoc Networks), introduces significant security challenges that must be addressed to ensure reliable and safe communications.

The principal objective of this research is to address these challenges by integrating cryptographic solutions that ensure the integrity, confidentiality, and authenticity of vehicular network messages. This thesis employs the use of Public Key Infrastructure (PKI) and the Elliptic Curve Digital Signature Algorithm (ECDSA) as integral components in order to effectively address and mitigate potential threats such as message tampering, impersonation, and eavesdropping. This approach enhances the real-time trustworthiness of vehicle-infrastructure communication.

A significant contribution of this research is the implementation of security protocols based on the IEEE 1609.2 standard, which encompasses encryption, digital signatures, and certificate management. The deployment of digital signatures enables the system to ascertain the veracity of messages, which is a vital component in enabling only authenticated vehicles to gain access to the network. Furthermore, this thesis employs ProVerif for formal verification, meticulously examining security properties such as reachability, secrecy, and authentication to ascertain that the implemented cryptographic measures effectively mitigate identified vulnerabilities in dynamic vehicular settings.

The efficacy of these security protocols is corroborated through simulations conducted within the MS-VAN3T and OScar frameworks, wherein latency and packet reception ratios (PRR) are quantified. The results demonstrate that the security measures have a negligible impact on system performance, even under high traffic conditions. This indicates that these protocols are suitable for implementation in real-world V2X systems. The final stage of the thesis involved a connection with

the PKI system in order to obtain the necessary EC and subsequently the AT, which would then be responsible for gathering the valid certificate and incorporating it into the V2X message.

In conclusion, this thesis puts forth a comprehensive solution for V2X security, integrating sophisticated cryptographic techniques and validating their effectiveness through formal methods and simulations. While successfully enhancing authentication and integrity, future work should prioritize addressing privacy concerns, such as pseudonym management, to safeguard user data on identity and location. These enhancements would complete a comprehensive security framework that meets the full spectrum of modern V2X security requirements.



# Acknowledgements

I would like to express my gratitude to my supervisors, Claudio Ettore Casetti and Riccardo Sisto, and to all those who provided assistance and support throughout this process, including Marco Rapelli, Simone Bussa, Carlos Mateo Risma Carletti, and Francesco Raviglione. Their availability, kindness, and consistent encouragement were invaluable.

I am also grateful to my friends, who have been a constant source of companionship and encouragement during this journey.

I would like to express my gratitude to my girlfriend, Manuela, who has consistently provided me with emotional and moral support.

In conclusion, I must express my gratitude to my family, without whom the realisation of this ambitious objective would not have been possible.



# Table of Contents

<b>List of Figures</b>	IX
<b>Acronyms</b>	XII
<b>1 Introduction</b>	1
1.1 Overview of V2X Communication . . . . .	1
1.1.1 Historical Review . . . . .	3
1.1.2 The Emergence and Evolution of IEEE 802.11p . . . . .	4
1.1.3 The Emergence of Cellular-Based V2X (C-V2X) . . . . .	6
1.2 Security in V2X Communication . . . . .	8
1.2.1 VANET Security Services . . . . .	9
1.2.2 Main Types of Attacks . . . . .	10
1.2.3 Solutions to Address Security Threats . . . . .	14
1.3 Structure of the Thesis . . . . .	17
<b>2 Standards</b>	19
2.1 Intelligent Transport System (ITS) architecture . . . . .	19
2.1.1 ITS application groups . . . . .	23
2.2 Geonetworking Protocol . . . . .	24
2.2.1 GeoNetworking Packet Structure . . . . .	25
2.2.2 Wireless Access in Vehicular Environments (WAVE) . . . . .	28
2.2.3 Security services protocol data units (SPDUs) . . . . .	29
2.2.4 Certificate Revocation Method . . . . .	32
2.3 ITS Security Management System . . . . .	33
2.3.1 ITS station security lifecycle . . . . .	36
2.3.2 Cryptographic operations . . . . .	37
2.3.3 Enrolment Management . . . . .	38
2.3.4 Authorization Management . . . . .	40
2.4 Security profiles and certificate formats . . . . .	42
2.4.1 CAM profile . . . . .	43

<b>3</b>	<b>Formal Verification</b>	45
3.1	Introduction . . . . .	45
3.1.1	Model checking . . . . .	45
3.1.2	Theorem Proving . . . . .	46
3.2	ProVerif . . . . .	46
3.2.1	PKI verification . . . . .	47
3.3	Verification Results . . . . .	48
<b>4</b>	<b>ms-van3t</b>	53
4.1	Introduction . . . . .	53
4.2	Security Header . . . . .	55
4.2.1	ASN.1 Compilation and Integration into ms-van3t . . . . .	55
4.2.2	CAM profile structure and message verification . . . . .	55
4.2.3	Digital Signature: Creation and Verification . . . . .	57
4.3	Simulation results . . . . .	60
4.3.1	Average Latency . . . . .	61
4.3.2	Packet Reception Ratio (PRR) . . . . .	63
4.3.3	Analysis . . . . .	64
<b>5</b>	<b>OScar</b>	66
5.1	Code Porting to the OScar Framework . . . . .	66
5.2	Test Methodology . . . . .	67
5.2.1	Field Testing Setup . . . . .	67
5.2.2	Server Testing Setup with GPS Trace Replay . . . . .	70
5.3	Results . . . . .	70
5.3.1	Metrics and Formulas Used . . . . .	70
5.3.2	Latency Data from Server and Field Tests . . . . .	71
5.3.3	Analysis of Latency and Performance Impact of CPU Differences . . . . .	71
5.4	PKI Integration and Real Certificate Management . . . . .	72
<b>6</b>	<b>Conclusion</b>	75
6.1	Key contributions . . . . .	75
6.2	Future works . . . . .	76
<b>A</b>	<b>Proverif C-ITS trust model</b>	78
A.1	Standard Scheme of C-ITS Trust Model . . . . .	78
A.1.1	Public and Private Key Definitions . . . . .	78
A.1.2	Request Functions . . . . .	78
A.2	Public-Key Cryptography . . . . .	79
A.2.1	Encryption and Decryption . . . . .	79
A.3	Processes in the Trust Model . . . . .	79
A.3.1	AA Process . . . . .	79

A.3.2	EA Process . . . . .	79
A.3.3	ITS-S Process (Initiating Request) . . . . .	80
A.4	Security Queries . . . . .	81
A.4.1	Reachability Queries . . . . .	81
A.4.2	Secrecy and Authentication Queries . . . . .	81
A.4.3	Event and Query Extensions for AT Provisioning . . . . .	81
<b>B</b>	<b>Security Header</b> . . . . .	<b>82</b>
B.1	GeoNetworking Integration . . . . .	82
B.1.1	CreateSecurePacket . . . . .	82
B.1.2	CreateSecurePacket Function Definition . . . . .	82
B.1.3	Certificate or Digest Selection Logic . . . . .	84
B.1.4	ExtractSecurePacket . . . . .	84
B.1.5	ExtractSecurePacket Function Definition . . . . .	85
B.2	Signature Algorithm . . . . .	87
B.2.1	Cryptographic Functions for Signature . . . . .	87
B.2.2	Signing Hashes . . . . .	89
B.3	Signature Creation . . . . .	90
B.4	Signature Verification . . . . .	91
B.5	Encryption Algorithm for requests to Authorities . . . . .	93
B.5.1	Cryptographic Functions for Encryption . . . . .	93
B.5.2	Encryption Process . . . . .	95
	<b>Bibliography</b> . . . . .	<b>98</b>

# List of Figures

1.1	Overview of V2X Communication System . . . . .	1
1.2	Protocol stack for ETSI ITS / 802.11 OCB . . . . .	6
1.3	VANET security services and related attacks . . . . .	10
1.4	Scenario for Sybil attack effect . . . . .	12
1.5	Impact of Sybil attack on VANET performance . . . . .	13
1.6	Digital signature mechanism . . . . .	16
2.1	ITS station reference architecture/ITS-S host . . . . .	20
2.2	GeoUnicast communication . . . . .	25
2.3	GeoBroadcast communication . . . . .	25
2.4	GeoNetworking Header . . . . .	27
2.5	GeoNetworking with security . . . . .	28
2.6	WAVE protocol stack showing detail of WAVE Security Services . . . . .	28
2.7	Security header SPDUs . . . . .	32
2.8	Logic flows used in determining the CRACA for a certificate . . . . .	33
2.9	PKI architecture . . . . .	34
2.10	PKI architecture with butterfly AT provisioning . . . . .	35
2.11	Butterfly key expansion . . . . .	36
2.12	ITS Station Security Life Cycle . . . . .	37
2.13	Enrolment Request structure . . . . .	39
2.14	Enrolment Response structure . . . . .	40
2.15	Authorization Request structure . . . . .	41
2.16	Authorization Response structure . . . . .	42
3.1	Sequence to achieve signed message transfer between ITS-Ss . . . . .	48
3.2	Reachability property result . . . . .	49
3.3	Secrecy property result . . . . .	50
3.4	Authentication property result - non-injective correspondence . . . . .	51
3.5	Authentication property result with Butterfly AT provisioning - non-injective correspondence . . . . .	51
3.6	Non-interference property result . . . . .	52

4.1	Full architecture of ms-van3t . . . . .	54
4.2	Packet capture via wireshark . . . . .	57
4.3	Pseudocode signature creation . . . . .	59
4.4	Pseudocode signature verification . . . . .	60
4.5	ms-van3t simulation scenario . . . . .	61
4.6	Average latency as a function of vehicle density . . . . .	62
4.7	Packet Reception Ratio (PRR) as a function of vehicle density . . . . .	63
5.1	OBU installed in vehicle . . . . .	68
5.2	TX and RX of CAMs . . . . .	68
5.3	Power Station . . . . .	68
5.4	Antenna . . . . .	68
5.5	Wireshark capture of CAMs . . . . .	69
5.6	CAM with SignerIdentifier equal to certificate . . . . .	69
5.7	CAM with SignerIdentifier equal to digest . . . . .	70





# Acronyms

**ITS**

intelligent transport system

**C-ITS**

Cooperative Intelligent Transport Systems

**ETSI**

European Telecommunications Standards Institute

**IEEE**

Institute of Electrical and Electronics Engineers

**V2X**

Vehicle to Everything

**V2I**

Vehicle to Infrastructure

**V2P**

Vehicle to Pedestrian

**V2V**

Vehicle to Vehicle

**NHTSA**

National Highway Traffic Safety Administration

**DSRC**

Dedicated Short-Range Communication

**WAVE**

Wireless Access in Vehicular Environment

**VANET**

Vehicular ad Hoc Networks

**PKI**

Public Key Infrastructure

**CAM**

Cooperative Awareness Message

**DENM**

Decentralized Environmental Notification Message

**GN**

GeoNetworking

**CA**

Certification Authority

**CRLSeries**

Certificate Revocation List Series

**CRACA**

Certificate Revocation Authorizing CA

**EC**

Enrolment Certificate

**EA**

Enrolment Authority

**AA**

Authorisation Authority

**AT**

Authorisation Ticket

# Chapter 1

# Introduction

## 1.1 Overview of V2X Communication

The automotive industry is undergoing a substantial transformation as a result of the integration of sophisticated communication technologies, which enable vehicles to interact with their surrounding environment. The exchange of information between vehicles (V2V), infrastructure (V2I), pedestrians (V2P), and networks (V2N) is made possible by Vehicle-to-Everything (V2X) communication, as illustrated in Figure 1.1.

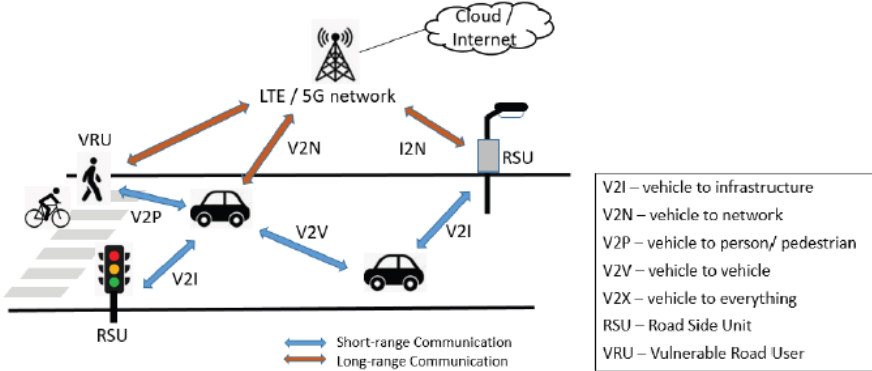


Figure 1.1: Overview of V2X Communication System

The primary objective of V2X is to enhance road safety, optimise traffic flow, and facilitate autonomous driving through the real-time transmission of data but also to improve traffic efficiency and reduce environmental impacts. The National Highway Traffic Safety Administration (NHTSA) in the United States estimates that the implementation of Vehicle-to-Vehicle (V2V) technology could prevent between 439,000 and 615,000 accidents annually. Furthermore, this could result in

a reduction of 987 to 1,366 fatalities and 537,000 to 746,000 incidents of property damage per year, which serves to illustrate the considerable potential of V2X to enhance road safety and reduce the incidence of accidents.

In Europe, the European Commission has recognised the significant benefits of Cooperative Intelligent Transport Systems (C-ITS). These systems promise to improve transport efficiency by reducing journey times and accident rates through improved communication between vehicles. In addition, C-ITS aims to reduce fuel consumption by reducing congestion and improving vehicle flow. One of the enabling technologies for these systems is based on the IEEE 802.11p standard, which facilitates communication between vehicles in what is known as "Outside the Context of a Basic Service Set" (OCB) mode. This allows vehicles to communicate directly with each other and with roadside infrastructure without the need for a centralised network.

A similar technology has been developed in the United States under the name Dedicated Short-Range Communication (DSRC). This system is governed by several standards, including the IEEE 1609 series and SAE J2735, which together form the Wireless Access in Vehicular Environments (WAVE) framework. WAVE is responsible for defining the communication protocols used between vehicles and roadside infrastructure. In Europe, the counterpart to DSRC is ITS-G5, which is also based on the IEEE 802.11 standard in OCB mode. ITS-G5 has been integrated into the broader European C-ITS framework, ensuring interoperability of vehicle communication systems across countries and manufacturers. This standardisation is key to the widespread adoption of vehicle-to-everything (V2X) technology, enabling vehicles from different manufacturers and across borders to communicate seamlessly with each other and with roadside units.

The progress of V2X technology relies heavily on collaboration between regulators, industry players and researchers. Several organisations are involved in developing the necessary standards and guidelines to ensure the safety and reliability of these systems. Notable contributors include ASTM International, the Institute of Electrical and Electronics Engineers (IEEE), the European Telecommunications Standards Institute (ETSI), the Society of Automotive Engineers (SAE) and the Third Generation Partnership Project (3GPP). These organisations are working with government agencies to establish the framework that will govern V2X communications in the future. For example, 3GPP plays a key role in setting standards for cellular-based V2X systems, while IEEE continues to advance wireless communication standards through initiatives such as the IEEE 802.11bd working group, which aims to improve vehicle communication technologies.

Government involvement is also critical to the advancement of V2X technology. In the United States, the Department of Transportation (USDOT) has launched several initiatives to enhance V2X capabilities and facilitate its integration into the national transportation network. Projects such as Traffic Optimisation for

Signalised Corridors (TOSCo), Cooperative Automated Driving Systems (CADS), and Vehicle-to-Infrastructure Safety Applications (V2I-SA) exemplify the government's commitment to the development and deployment of V2X technologies. These initiatives focus on improving traffic management, safety and coordination of automated driving systems.

In Europe, numerous projects have received financial support from the European Union, in particular through programmes such as 5G-DRIVE, MARSS-5G and C-ROADS. In addition, Horizon Europe, the European Union's research and innovation programme running from 2021 to 2027, has supported V2X-related projects such as SwiftV2X and 5GMED. These projects are at the forefront of innovation, addressing the intersection of the automotive and telecommunications sectors and driving the development of next-generation V2X systems.

In Asia, Japan has made significant progress in developing V2X and related technologies through a combination of government-led initiatives and research programmes. Services such as the Vehicle Information and Communication System (VICS), Electronic Toll Collection (ETC), Smartway and ITS Spot Service have been implemented, providing real-world examples of V2X in action. In addition, Japan's Strategic Innovation Promotion (SIP) programme for automated driving, known as Automated Driving for Universal Services (SIP-adus), promotes collaboration between academia, industry and government. This initiative aims to accelerate the development of automated driving technologies and establish a robust framework for V2X communications in the country [1].

In conclusion, advances in vehicle-to-everything (V2X) communications, driven by advancements in standards, research initiatives and government programmes, will bring profound changes to the automotive industry. The integration of V2X technology promises not only to improve road safety, but also to improve the overall efficiency of transport systems, reduce congestion and reduce the environmental impact of vehicles. By enabling vehicles to communicate seamlessly with each other and with roadside infrastructure, V2X will transform the way transport networks operate, making them safer, smarter and more sustainable.

### 1.1.1 Historical Review

The concept of vehicles communicating with each other and with the infrastructure, now known as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, dates back to the 1970s. At that time, the primary motivation was to improve road safety and reduce the number of road accidents. Early initiatives such as the Electronic Route Guidance System (ERGS) in the United States and the Comprehensive Automobile Traffic Control System (CACCS) in Japan laid the foundations for more advanced systems. These projects were among the first to explore how vehicles could share information to optimise route planning and improve

traffic management. In the early 1980s, research into V2V communication began to intensify. Different types of media were experimented with, including infrared signals, very high frequency (VHF) radio and millimetre waves. Communication protocols such as ALOHA and Carrier Sense Multiple Access (CSMA) were used to manage the flow of information between vehicles. Several major research projects emerged during this period, most notably the European PROMETHEUS initiative and the US PATH and NAHSC programmes. These early efforts focused on the potential of technology to support vehicle-to-vehicle communication, whether through direct vehicle-to-vehicle interactions or through infrastructure.

At the turn of the century, the term 'vehicular ad hoc network' (VANET) was introduced, reflecting the adaptation of mobile ad hoc network (MANET) concepts to the vehicular environment. In this context, VANET became synonymous with Inter-Vehicle Communication (IVC), referring to both vehicle-to-vehicle and vehicle-to-infrastructure communication. This terminological shift signalled a growing recognition of the potential of autonomous communication networks in transport. Global initiatives such as Japan's Advanced Safety Vehicle (ASV), Europe's SAFESPOT and PReVENT, and the United States' COMeSafety further illustrated the international momentum behind V2X technology. These projects reflected a collective effort to explore how vehicles could communicate not only with each other, but also with the surrounding infrastructure to improve road safety and transport efficiency.

In the early stages of V2X development, a variety of communication media were explored, ranging from infrared to radio waves and even GPS signals. Over time, certain technologies became dominant and the V2X communication landscape became more standardised. Concepts such as Dedicated Short-Range Communication (DSRC), Wireless Access in Vehicular Environments (WAVE) and the Internet of Vehicles (IoV) entered the discourse. Standards such as IEEE 802.11p and its European counterpart, ITS-G5, also gained widespread recognition. These standards were instrumental in defining how vehicles could connect to each other and to infrastructure in real time, paving the way for more advanced and reliable communication systems. Today, the dominant technologies in the V2X space are cellular networks, specifically those described in 3rd Generation Partnership Project (3GPP) Release 16, and WiFi-based communications using the IEEE 802.11p standard. These technologies enable connected vehicles to communicate with each other, roadside infrastructure and even pedestrians, offering the potential for safer and more efficient transport systems on a global scale [2].

### **1.1.2 The Emergence and Evolution of IEEE 802.11p**

The introduction of the IEEE 802.11p standard in the early 2000s marked a major milestone in the evolution of vehicular communications. Known as Wireless Access

in Vehicular Environments (WAVE), this standard was specifically designed to meet the demands of high-speed, low-latency communications in the dynamic environment of vehicular networks. Operating in the 5.9 GHz frequency band allocated by the Federal Communications Commission (FCC) for Dedicated Short-Range Communication (DSRC) [3], IEEE 802.11p introduced significant improvements over traditional wireless technologies.

Unlike conventional WiFi, which is challenged by high mobility and changing topologies, IEEE 802.11p is tailored to address these specific issues. It enables critical applications such as collision avoidance, traffic coordination and the dissemination of real-time information between vehicles and infrastructure, directly supporting road safety and efficiency.

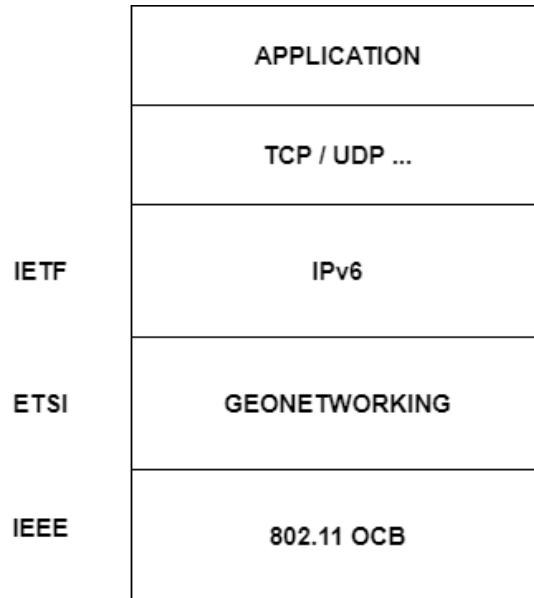
One of the key features of IEEE 802.11p is its ability to establish communications without the need for prior authentication or association, thanks to its Outside the Context of a Basic Service Set (OCB) mode. This allows vehicles and roadside units to communicate directly, bypassing the delays typically associated with traditional WiFi connections. However, despite its technical advantages, IEEE 802.11p has been criticised for lacking built-in security mechanisms and relying on higher-layer protocols for security.

The responsibility for securing communications falls to standards such as IEEE 1609.2, which defines the use of encryption, authentication and certificate management via a Public Key Infrastructure (PKI). While this approach provides necessary security services, it also introduces challenges related to the complexity of certificate management and revocation, creating potential vulnerabilities in the system.

Recognising these security gaps, the European Telecommunications Standards Institute (ETSI) has developed a comprehensive framework as part of its Intelligent Transport Systems (ITS) initiative. ETSI's ITS standards address the specific security, privacy and interoperability needs of V2X communications across Europe [4]. These standards include several protocols and mechanisms designed to secure V2X communications:

- EN 302 636: Defines the GeoNetworking protocol to ensure efficient message dissemination.
- TS 102 940: Establishes a security architecture encompassing authentication, authorization, and privacy protection.
- TS 102 941: Focuses on trust and privacy management, ensuring that only authorized entities can access the V2X network.
- TS 102 942: Details access control mechanisms for preventing unauthorized use.

- TS 103 097: Defines security headers and certificate formats for secure message exchanges.



**Figure 1.2:** Protocol stack for ETSI ITS / 802.11 OCB

Together, these standards provide a robust security framework that addresses the vulnerabilities identified in IEEE 802.11p and ensures that V2X communications are properly encrypted, authenticated and protected from tampering, as illustrated in Figure 1.2.

### 1.1.3 The Emergence of Cellular-Based V2X (C-V2X)

While IEEE 802.11p formed the backbone of early V2X communications, the cellular industry began to develop its own solution: Cellular-based V2X (C-V2X). This technology uses the widespread infrastructure and capabilities of cellular networks to support V2X applications, providing an alternative to the WiFi-based IEEE 802.11p system.

#### **LTE-V2X**

Introduced in 3GPP Release 14, LTE-V2X (Long-Term Evolution for V2X) supports both direct communication between vehicles and communication through cellular networks. This is achieved via two interfaces: the PC5 interface, which enables direct communication between vehicles, and the Uu interface, which enables vehicles to communicate with the wider network via the cellular infrastructure.



Compared to IEEE 802.11p, LTE-V2X offers several key advantages, including higher data rates, better scalability and more reliable performance, particularly in dense urban environments. It also benefits from the mature security protocols of cellular networks, such as Authentication and Key Agreement (AKA), which are already widely used in mobile communications.

## 5G V2X

Building on the foundation of LTE-V2X, 5G V2X represents a significant leap forward in performance and capability. With ultra-low latency, higher data throughput and improved reliability, 5G V2X is tailored for more advanced use cases such as cooperative automated driving and remote vehicle control, which require robust and responsive communications.

In addition to improved performance metrics, 5G V2X introduces new capabilities such as network slicing and edge computing. These features allow the network to allocate resources more efficiently and process data closer to the source, enabling faster and more efficient communications for connected vehicles.

## Hybrid Approach and Future Directions

As both IEEE 802.11p and C-V2X have distinct advantages, the potential for a hybrid approach combining these technologies has gained increasing interest. Such a model would combine the strengths of IEEE 802.11p in short-range, low-latency communication with the wide-area coverage and enhanced capabilities of cellular-based V2X. This could result in a more versatile and effective V2X communication system that addresses a wider range of scenarios and use cases.

However, there are challenges to integrating these two systems. Achieving interoperability between the two technologies requires standardised protocols and interfaces to ensure seamless communication. Equally important is the harmonisation of their security frameworks to ensure that data is consistently protected across both systems. Standardisation bodies - such as ETSI and 3GPP - have a crucial role to play in addressing these issues to enable the smooth interworking of different V2X technologies.

In summary, the evolution of V2X standards underscores the ongoing effort to improve road safety, traffic efficiency and the overall performance of transportation networks. While IEEE 802.11p and related ETSI ITS standards provide a solid foundation for short-range communication, the development of LTE V2X and 5G V2X offer promising improvements in coverage and scalability. As these technologies evolve, the future of connected transport may lie in a hybrid approach that maximises the strengths of both WiFi-based and cellular-based V2X systems [4].

## 1.2 Security in V2X Communication

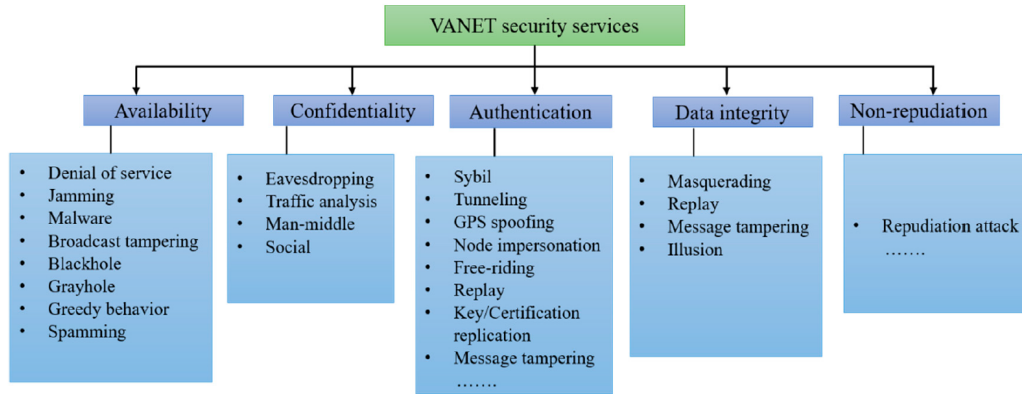
Vehicular ad hoc networks represent the foundation of V2X communications, a crucial element in contemporary transportation systems. The real-time data exchange facilitated by VANETs between vehicles and infrastructure renders them an indispensable tool for improving road safety, optimising traffic flow and preventing accidents. However, the reliance on open wireless communication, coupled with the dynamic nature of vehicular mobility, gives rise to a number of security challenges. Furthermore, the rapid advancement of vehicle technology has resulted in a notable increase in the number of electronic control units (ECUs) and a substantial rise in the complexity of the software utilized in modern vehicles. The number of electronic control units in standard vehicles is in excess of 80, rising to as many as 150 in luxury models. The software code in these vehicles exceeds 100 million lines, representing a significant increase in complexity. This growing complexity expands the digital attack surface, exposing vehicles to increased security and privacy risks. To illustrate, the Controller Area Network (CAN) bus, a foundational communication network within vehicles, is devoid of intrinsic security features such as authentication and message integrity. Furthermore, it operates under the assumption of a threat-free environment [1]. This omission is a significant cause for concern, particularly in the context of the potential impact of cyber-attacks on road safety and national infrastructure as well as the decentralised structure of VANETs, which lacks a central authority to monitor or control communications, which renders the entire system highly vulnerable to a range of cyber-attacks. It is of the utmost importance to guarantee the security of V2X systems. In consideration of the increasing prevalence of autonomous and semi-autonomous vehicles, it is of paramount importance to ensure the security and dependability of vehicular communication systems. A single compromised node in a VANET has the potential to disseminate erroneous data, which could result in adverse consequences such as accidents or significant disruptions to traffic flow. Moreover, the high mobility of vehicles, which can rapidly enter and exit communication zones, serves to exacerbate the vulnerability. It is relatively straightforward for malicious actors to exploit the ephemeral connections between vehicles to inject false information, manipulate traffic flows, or deny essential services. Despite the critical role of V2X communication, these networks are susceptible to a multitude of threats. The intrinsic nature of open and decentralised communication renders it vulnerable to message tampering, whereby attackers can alter or inject false data. This could result in misleading traffic information, which could then cause dangerous decisions to be made by autonomous systems or human drivers. Furthermore, the frequent disconnection and reconnection of vehicles in the network increases the attack surface, allowing malicious entities more opportunities to disrupt the network [5].

### 1.2.1 VANET Security Services

In order to address these identified vulnerabilities, it is essential to incorporate robust security services within VANETs. The objective of these services is to protect the network against potential attacks while maintaining uninterrupted communication. In this context, five key security properties are of paramount importance: availability, confidentiality, authentication, data integrity, and non-repudiation:

- **Availability** is arguably the most fundamental security requirement for VANETs. It is imperative that the network remains operational and accessible, even in the face of adversarial actions such as denial-of-service (DoS) attacks or jamming. In the event of a failure of the network to deliver safety messages in a timely manner, the consequences could be catastrophic, such as vehicles failing to receive critical warnings about road hazards or traffic conditions.
- **Confidentiality** is the assurance that sensitive information, such as a vehicle's location or identity, is accessible solely to authorised parties. This property is of particular significance in the prevention of eavesdropping and other forms of data interception. Cryptographic techniques, such as encryption, are frequently utilised to ensure confidentiality, thereby safeguarding the privacy of the driver and preventing unauthorised access to sensitive data.
- **Non-repudiation** provides assurances that neither the sender nor the receiver of a message can refute their involvement in the communication. This property is crucial for maintaining accountability, particularly in instances where disputes arise—such as a vehicle disputing the transmission of a critical safety message after an accident.
- **Data integrity** is fundamental to the assurance that the information transmitted across a network is not altered or tampered with during the transmission process. In the context of vehicular ad hoc networks (VANETs), the maintenance of data integrity is of paramount importance, as any alteration to the data could potentially lead to dangerous decisions being made. The utilisation of cryptographic methods, such as digital signatures, serves to guarantee that the data remains unaltered from its source to its intended destination.
- **Authentication** plays a crucial role in VANETs to ascertain the genuine identities of the communicating vehicles. Without robust authentication mechanisms, there is a risk of attackers impersonating legitimate vehicles, which could potentially result in traffic disruption and the dissemination of misleading information. As an illustrative example, authentication plays a

pivotal role in preventing Sybil attacks, a type of manipulation in which a single vehicle creates numerous false identities to gain undue influence over the network.



**Figure 1.3:** VANET security services and related attacks

## 1.2.2 Main Types of Attacks

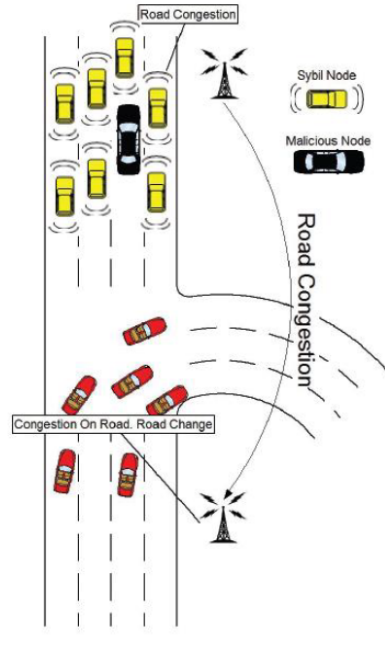
A variety of attacks targeting the core security properties of VANETs have been identified, underscoring the critical need to develop a comprehensive understanding of these threats in order to devise effective defence mechanisms. As the use of vehicular ad hoc networks (VANETs) increases in conjunction with the advent of autonomous and semi-autonomous vehicles, it is of paramount importance to ensure the security of these systems. The security of VANETs is contingent upon the provision of several key services, including availability, confidentiality, authentication, data integrity and non-repudiation. The security services in question are indispensable for the secure and reliable operation of the network; however, they are also susceptible to a multitude of sophisticated attacks.

- One of the most significant and pervasive threats to the availability of VANETs is the **denial-of-service (DoS)** attack. In these attacks, malicious vehicles or external adversaries inundate the network with an excessive, and often false, number of requests, thereby saturating the communication channels and overwhelming the system. Consequently, legitimate messages, including those of a critical safety nature, cannot be transmitted or received in a timely manner. The consequences of a DoS attack can be catastrophic, particularly in situations where real-time communication is vital for road safety, such as in

emergency braking systems, collision avoidance, or traffic rerouting. A more potent variant of this attack is the distributed denial-of-service (DDoS) attack, which employs the coordinated actions of multiple attackers or compromised devices to disrupt communication over a vast geographical area. Such a coordinated assault has the potential to bring entire sections of a vehicular network to a standstill, resulting in significant disruption and, in extreme cases, directly endangering road safety.

- A further form of attack that targets availability is the **Jamming attack**, which disrupts the physical communication channels utilised by VANETs. Adversaries introduce a substantial amount of noise on the same frequencies utilized for vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication. Such interference effectively blocks or distorts legitimate signals, thereby preventing vehicles from receiving essential updates, such as warnings about accidents, road conditions, or traffic jams ahead. The potential for remote execution of jamming attacks renders them particularly dangerous, as the attacker need not be physically present at the target location. This renders the detection and mitigation of the source of the interference in real time challenging, thereby further exacerbating the vulnerability of the network.
- Breaches in confidentiality typically manifest as **Eavesdropping**. It involves the unauthorised interception and listening in on communications between vehicles, allowing attackers to extract sensitive information such as the location of vehicles, driver identities, and even vehicle-specific data such as speed or fuel consumption. Such information can then be exploited for a variety of malevolent purposes, including the tracking of vehicle movements, the compromise of driver privacy, or the utilisation of the data for further attacks on the network.
- **Traffic analysis** represents a more passive form of attack, whereby adversaries study communication patterns over time with a view to gleaning insights into the operation of the network. By analysing the frequency and volume of data exchanges, attackers can infer valuable information about traffic flows, vehicle density in specific areas and potential vulnerabilities in the network's structure. While traffic analysis may not involve direct interference with the data being transmitted, it can still provide attackers with sufficient intelligence to enable them to carry out more targeted and effective attacks.
- One of the most significant and detrimental threats to authentication within vehicular ad hoc networks (VANETs) is the **Sybil attack**. In this attack, a single malevolent entity fabricates multiple false identities, effectively impersonating several distinct vehicles within the network. The creation of this illusion of multiple vehicles allows the attacker to manipulate the flow

of traffic information and deceive legitimate vehicles into taking erroneous actions, as illustrated in Figure 1.4. To illustrate, an adversary could simulate a congestion scenario by disseminating multiple messages from disparate fictitious identities, thereby deceiving proximate vehicles into rerouting to evade nonexistent congestion. Similarly, a Sybil attacker could influence the decisions made by autonomous systems by introducing false data about road conditions, which could result in delays, inefficiencies, or even accidents.

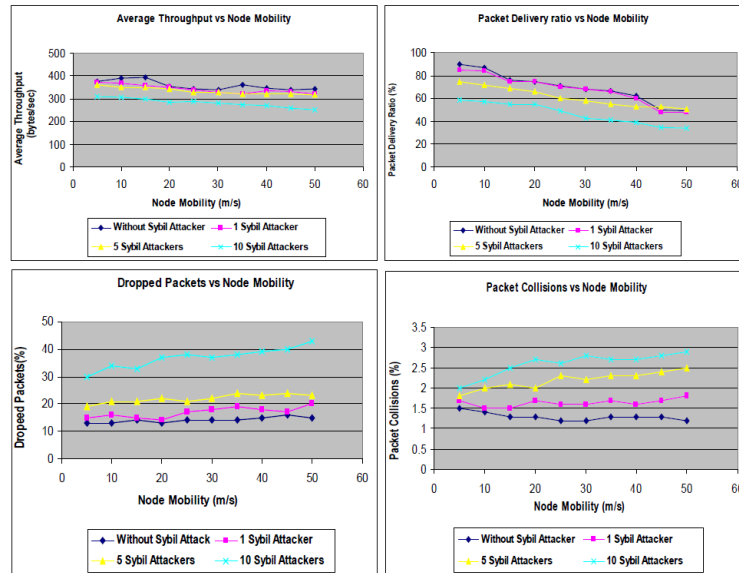


**Figure 1.4:** Scenario for Sybil attack effect

The capacity to create a multitude of identities affords attackers considerable control over the network, rendering Sybil attacks a particularly perilous threat. It is imperative that authentication systems are designed in a manner that ensures the reliable verification of each vehicle's identity, thus enabling the detection and prevention of such attacks [6]. The very nature of Sybil attacks poses a direct threat to the integrity and reliability of vehicular communications. The creation of multiple fake identities undermines the trustworthiness of the data being exchanged, allowing attackers to sow confusion and disrupt normal traffic operations. The decentralised nature of VANETs renders them particularly susceptible to this type of attack, given that there is often no central authority continuously monitoring the identities of all vehicles on the network. This lack of oversight enables attackers to generate as many fake identities as they wish, thereby greatly amplifying the potential damage of a

Sybil attack .

A study conducted by researchers from the Malaviya National Institute of Technology in Jaipur, India, evaluated the impact of Sybil attacks on the performance of vehicular ad hoc networks (VANETs) through simulations. The findings demonstrate that as the number of Sybil attackers increases, there is a notable decline in the network's performance. As illustrated in the accompanying graph, the packet delivery ratio declines to approximately 30% when 10 Sybil attackers are introduced, while the throughput decreases from 500 bytes per second in the absence of attackers to a mere 150 bytes per second. Furthermore, the number of dropped packets increases significantly, reaching over 70% as node mobility rises, and packet collisions reach approximately 3% [7]. These quantitative results demonstrate the considerable impact that Sybil attacks can have on the reliability and efficiency of vehicular communication systems, as can be seen in Figure 1.5.



**Figure 1.5:** Impact of Sybil attack on VANET performance

Consequently, effective authentication mechanisms are critical in preventing the network from being flooded with false data.

- Other attacks on authentication include **tunneling**, whereby attackers create false links between disparate parts of the network, thereby causing vehicles to appear to be situated in locations that they are not. By manipulating the network topology, it is possible for attackers to deceive vehicles into making incorrect routing decisions, thereby further complicating traffic management

and potentially leading to dangerous situations on the road.

- further significant concern for VANETs is the potential for attacks on data integrity. In **message tampering attacks**, adversaries modify the contents of messages as they are transmitted between vehicles. To illustrate, an adversary could modify a message indicating traffic congestion, thereby providing false information and misleading vehicles into navigating towards a congested area or one posing a hazard. The potential consequences of this type of attack are significant, as vehicles and drivers rely on accurate and timely information to make decisions about their routes and driving behaviour. By modifying these messages, attackers can create confusion, impede emergency response times, and precipitate dangerous driving conditions.
- Finally, **non-repudiation attacks**, such as repudiation, occur when a sender denies having sent a particular message, thereby making it challenging to ascertain responsibility or resolve disputes in the event of an incident. The non-repudiation of actions within a vehicular ad hoc network (VANET) is of paramount importance for maintaining accountability and ensuring the traceability of all actions taken within the network. When attackers are able to deny their involvement in sending malicious messages, it becomes significantly more challenging to identify and mitigate the source of the problem, leaving the network vulnerable to continued exploitation.

### 1.2.3 Solutions to Address Security Threats

In order to address the numerous security challenges that are inherent to VANETs, a number of solutions have been developed. It is evident that cryptographic techniques play a pivotal role in ensuring the security of communication within these networks. Cryptography provides the basis for the security architecture in VANETs, ensuring the authentication, confidentiality and protection from tampering of messages [5].

- Symmetric cryptography represents one of the most prevalent techniques employed for the protection of communication in VANETs. This approach relies on the use of a shared secret key between the communicating parties, which is employed for both the encryption and decryption of messages. The principal advantage of symmetric cryptography is its computational efficiency, which renders it particularly well-suited to real-time applications where rapid processing is essential. Nevertheless, symmetric cryptography is not without its limitations, particularly in regard to non-repudiation. The use of a single key by both the sender and the receiver makes it challenging to ascertain with absolute certainty which party originated a given message. This lack of accountability can prove problematic in scenarios where it is important to



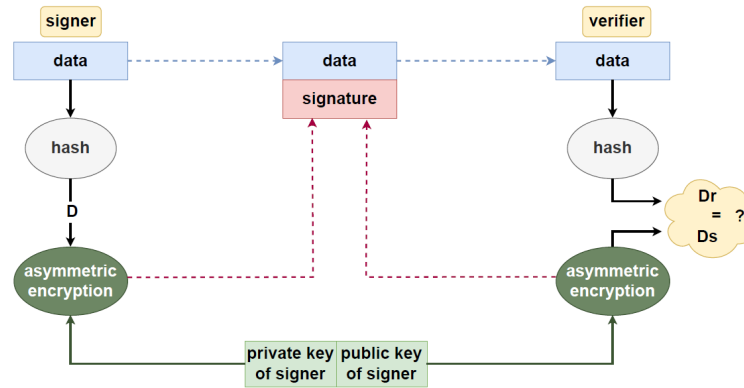
verify the identity of the message sender, particularly in cases of dispute or security breach.

- In contrast, asymmetric cryptography, often leveraged by systems like public key infrastructure (PKI), provides stronger guarantees of security by utilising a pair of cryptographic keys – one public and one private – for each vehicle. In this system, vehicles utilise their private key to sign messages, and the recipient can verify the signature using the corresponding public key. This ensures both authentication and non-repudiation, as the signature can be used to prove the origin of the message. Although PKI provides a more robust security framework, it comes with higher computational costs and requires more memory, which can present a challenge in resource-constrained environments such as vehicles with limited processing power and memory capacity.
- A more efficient alternative to traditional public key cryptography is the Elliptic Curve Digital Signature Algorithm (ECDSA). The security afforded by ECDSA is comparable to that of other public key systems; however, the shorter key lengths result in a significant reduction in the computational load on vehicles. This makes ECDSA an especially suitable choice for use in Vehicular Ad-hoc Networks (VANETs), where real-time communication is essential, and delays must be minimized.

One of the principal advantages of ECDSA is its capacity to provide a robust defense against Sybil attacks. The use of digital signatures to verify the identity of each vehicle ensures that only authenticated vehicles can participate in the network, thereby preventing attackers from generating multiple fake identities.

As illustrated in the Figure 1.6, the ECDSA process begins with the sender (or signer) hashing the message and encrypting the hash using their private key to generate a digital signature. This signature is transmitted alongside the original data. The recipient (or verifier) applies the same hash function to the received data and uses the sender's public key to decrypt the signature, revealing the original hash. If the decrypted hash ( $D_r$ ) matches the computed hash ( $D_s$ ), the signature is valid, proving both authentication and integrity. This ensures that the message has not been altered and that it originated from a legitimate source.

This thesis explores the implementation of digital signatures (DS) using ECDSA, alongside public key infrastructure (PKI), as a means of overcoming the challenges posed by authentication attacks. By focusing on digital signatures, it is possible to guarantee that each vehicle in the network possesses



**Figure 1.6:** Digital signature mechanism

a distinctive, verifiable identity, which is essential for maintaining trust and security in VANET communications.

- In addition to cryptographic solutions, privacy-preserving mechanisms such as pseudonym-based authentication play a vital role in protecting the identities of drivers while ensuring secure communication. In this approach, vehicles frequently change their pseudonyms, which makes it difficult for attackers to track them over time. This method strikes a balance between preserving privacy and maintaining accountability, as trusted authorities can still revoke pseudonyms in the event of malicious behaviour. The frequent rotation of pseudonyms helps to prevent long-term tracking of vehicles, thereby enhancing the privacy of drivers while still allowing for secure and reliable communication within the network.
- It is also the case that non-cryptographic solutions contribute to the security of vehicular ad hoc networks (VANETs). For instance, platooning systems permit vehicles to form collaborative groups that exchange data and work together to enhance security. In a platoon, vehicles communicate directly with one another, thereby enabling the sharing of information regarding their respective positions, speeds, and driving intentions. By working together in this way, vehicles can more readily identify anomalies or suspicious behaviour, thereby enhancing overall network security. Similarly, trust management frameworks evaluate the behaviour of vehicles over time, allowing the network to detect and isolate malicious actors. In these models, vehicles build trust based on their past behaviour, and those with a history of malicious or suspicious activity are excluded from the network. This approach allows for more dynamic and adaptive security measures, reducing the network's reliance on cryptographic

solutions while still providing robust protection against attacks.

## 1.3 Structure of the Thesis

This thesis is comprised of six chapters, each of which focuses on a specific aspect of V2X communication and security, particularly in the context of vehicular ad hoc networks (VANETs).

- **Chapter 1: Introduction** This chapter provides an overview of V2X communication, outlining its evolution and the key standards that underpin it, such as IEEE 802.11p and C-V2X. It also addresses the security concerns associated with V2X, presenting the primary types of attacks and the security services that are required to mitigate them. This chapter establishes the context for the remainder of the thesis, outlining the motivation for exploring secure communication in VANETs.
- **Chapter 2: Standards** This chapter presents a discussion of relevant ETSI and IEEE standards for V2X communication, with a particular focus on the ITS architecture, the Geonetworking Protocol, and Public Key Infrastructure (PKI). The role of these standards in ensuring secure and reliable communication among vehicles is emphasised, as they form the foundational layer for the security mechanisms examined in subsequent chapters.
- **Chapter 3: Formal Verification** This chapter presents an overview of formal verification methodologies employed to validate the security mechanisms implemented in VANETs. It introduces tools such as ProVerif, which are used to model and analyse the reachability, secrecy, and authentication properties of the system. The verification results presented in this chapter provide evidence for the robustness of the proposed security solutions.
- **Chapter 4: ms-van3t** This chapter introduces the ms-van3t simulation framework, which is used to simulate a variety of security scenarios in vehicular ad hoc networks (VANETs). It provides a detailed account of the implementation of security headers, signature creation and verification, and packet analysis. The results from the simulations are discussed in order to evaluate the performance and effectiveness of the security mechanisms.
- **Chapter 5: OScar** This chapter presents a discussion of the OScar framework, a pivotal component of the security implementation. It builds upon the work conducted in preceding chapters to provide a comprehensive security framework that integrates with V2X communication systems.

- **Chapter 6: Conclusion** The concluding chapter presents a synthesis of the thesis's principal findings and suggests avenues for future research. It identifies potential avenues for strengthening security mechanisms and highlights remaining challenges in the domain of vehicular ad hoc networks (VANET) security, thus providing a framework for subsequent investigations.

Furthermore, the appendices furnish comprehensive elucidations of the ProVerif C-ITS trust model and the particular encoding, signature creation, and verification mechanisms employed in the security header.

# Chapter 2

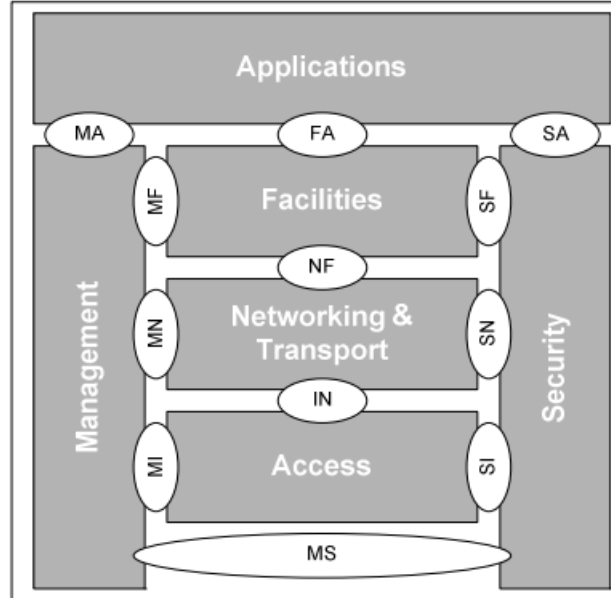
## Standards

The European Telecommunications Standards Institute (ETSI) occupies a pivotal position in the standardisation of V2X communications. As the scope of vehicular networks grows, standardisation assumes greater importance in guaranteeing interoperability, security and efficiency across a range of communication platforms. ETSI has developed a comprehensive set of standards for Intelligent Transport Systems (ITS), which provide the framework for V2X systems to operate seamlessly across different vehicle manufacturers, infrastructure providers, and regions. At the core of ETSI's work in V2X communications is the ETSI ITS-G5 standard, which is based on IEEE 802.11p as cited in 1.1.2 and governs short-range communications between vehicles and between vehicles and infrastructure (V2I). Furthermore, ETSI standards address the various communication layers and protocols, from the physical layer to the application layer, thereby ensuring the scalability and reliability of V2X services. Security is another significant area of focus for ETSI's V2X standards, addressing the necessity for privacy, integrity, authentication, and authorisation in vehicular networks. The ETSI standards define mechanisms for secure message exchange, the use of Public Key Infrastructure (PKI), and cryptographic techniques to protect V2X communications from malicious attacks. This section will provide a detailed overview of the ETSI standards relevant to V2X communication, including their technical aspects, implementation challenges, and their significance in the context of a secure, reliable, and scalable vehicular communication ecosystem.

### 2.1 Intelligent Transport System (ITS) architecture

The ITS architecture has been designed with the objective of supporting a wide range of applications, with the intention of improving traffic efficiency, safety and environmental impact. The architecture is comprised of multiple layers [8], each of

which serves a specific function to ensure robust and secure communication within the ITS environment.



**Figure 2.1:** ITS station reference architecture/ITS-S host

Here is an overview of the four horizontal layers:

- The Access Layer is responsible for the management of the physical and data link layers of communication within the Intelligent Transport System (ITS). It encompasses the technologies that enable direct communication between vehicles and between vehicles and infrastructure, over various media. The primary technologies involved in this layer include:
  - The IEEE 802.11p standard (Dedicated Short-Range Communications - DSRC) is employed for the purpose of facilitating short-range communication between vehicles and roadside units.
  - Cellular technologies, exemplified by Long Term Evolution (LTE) and 5G, facilitate extended-range communication and support high-data-rate applications.

The Access Layer ensures that messages are transmitted and received over the appropriate communication channels, managing issues such as signal interference, data collision, and link reliability.

- The Network and Transport Layer is of paramount importance for guaranteeing the correct and efficient delivery of data across the network. It is responsible for the routing and forwarding of messages, as well as the management of data flow between different ITS stations (vehicles, roadside units, central servers). The key functions of this layer include:
  - GeoNetworking: It is a protocol that has been specifically designed for the purpose of enabling communication between vehicles. It facilitates the routing of messages based on geographical location.
  - IPv6: It is a networking protocol that is used to handle addressing and routing, thereby enabling communication over broader networks, including the Internet.

The Network and Transport Layer is also responsible for managing congestion control and ensuring that data packets are transmitted with minimal delay, which is critical in time-sensitive applications such as safety warnings.

- The Facilities Layer provides indispensable support functions for Intelligent Transport Systems (ITS) applications. It serves as a middleware, enabling data exchange between the Network and Transport Layer and the Application Layer. The principal components of the Facilities Layer are as follows:
  - Message handling: This process is concerned with the receipt, transmission, and delivery of messages. It guarantees that messages are correctly processed and delivered to the relevant applications.
  - Data Management: This layer handles the storage, retrieval, and organisation of data, including the Local Dynamic Map (LDM), which provides a real-time representation of the vehicle's surrounding environment.
  - Service Discovery: This layer allows ITS stations to identify and utilise available services, such as traffic information or hazard warnings.

This layer plays a crucial role in ensuring that applications have access to accurate and timely data, which is essential for making informed decisions.

- The application layer is responsible for hosting the various ITS applications that provide services to end-users. The objective of these applications is to enhance traffic safety, efficiency and the overall user experience. Examples of Intelligent Transport Systems (ITS) applications include:
  - Transmission of messages on a periodic basis and messages triggered by specific events.
  - Traffic Management and Control: Applications that facilitate the optimization of traffic flow and the reduction of congestion.

- Infotainment Services: Provide information and entertainment to passengers, thereby enhancing their travel experience.

The application layer represents the interface between the user and the intelligent transport system (ITS), facilitating the receipt of information, alerts and other services that enhance driving safety and convenience.

Then we have other two vertical layer as we can see in figure 2.1:

- The Security Layer is of paramount importance in guaranteeing the integrity, authenticity, and confidentiality of communication within the ITS environment. In view of the critical nature of vehicular communications, this layer incorporates mechanisms to safeguard against unauthorised access, data manipulation, and other security threats. The key features of the Security Layer include:
  - Authentication and Authorization: Ensuring that only trusted entities can access and communicate within the ITS network.
  - Encryption: Protecting the confidentiality of messages by encrypting the data being transmitted.
  - Digital Signatures: Verifying the authenticity of messages and ensuring that they have not been tampered with.

The Security Layer operates across all horizontal layers, thereby ensuring that security is maintained throughout the communication process, from the physical transmission of data to its utilisation in applications.

- The Management Layer is responsible for the overall operation and coordination of the entire ITS architecture. Its functions include the configuration, monitoring, and maintenance of ITS stations and networks. The Management Layer ensures the efficient operation of the system and the prompt resolution of any issues that may arise. The key responsibilities of this layer include:
  - Network Management: Monitoring the status of the network, managing resources, and ensuring optimal performance.
  - Service Management: Overseeing the availability and performance of ITS services, ensuring they meet the required standards.
  - Configuration Management: Managing the settings and configurations of ITS stations to ensure they operate correctly.

The Management Layer facilitates the deployment of updates and patches, thereby ensuring that the ITS system remains current and secure against emerging threats.



### 2.1.1 ITS application groups

As outlined in reference [9], the application groups and associated messages form a fundamental part of the communication infrastructure for Intelligent Transport Systems (ITS), and the most important are:

1. **Cooperative Awareness (CA)** Cooperative Awareness applications are focused on enhancing road safety by sharing real-time information about a vehicle's current status with nearby vehicles and infrastructure.
  - Messages: Cooperative Awareness Messages (CAMs).
  - Objective: To share details such as the vehicle's speed, location, and movement direction to prevent accidents.
  - Key Characteristics:
    - Frequently broadcast.
    - Small data payloads.
    - Time-sensitive transmissions.
    - Enables both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication.
    - No need for a session to be established.

Examples include emergency vehicle warnings, intersection collision warnings, and merging traffic collision warnings.

2. **Static Local Hazard Warning** This group focuses on providing warnings from fixed roadside units regarding static road hazards. These messages are sent continuously to inform drivers about specific risks.
  - Purpose: To warn vehicles about static road conditions like roadworks or obstacles.
  - Key Characteristics:
    - Broadcast from roadside units
    - Single-hop
    - Time-critical
    - low-data content

Examples include warnings for wrong-way driving and stationary vehicles.

3. **Interactive Local Hazard Warning** This group involves two-way communication between vehicles and roadside units. A vehicle sends a hazard alert and receives a response aimed at managing or resolving the risk.

- Messages: Broadcasts followed by a unicast session between the communicating parties.
  - Objective: To manage coordinated responses to hazards, such as pre-crash warnings.
4. **Area Hazard Warning** These messages, known as Decentralized Environmental Notification Messages (DENMs), alert vehicles about an ongoing hazardous event (e.g., an accident or weather-related danger). They are triggered by specific events rather than being broadcast continuously.
- Messages: DENMs.
  - Objective: To notify surrounding vehicles about hazardous situations, improving road safety.
  - Key Characteristics:
    - Multi-hop, geocast transmissions
    - Low data content
    - Event-driven, time-sensitive

Examples include alerts for emergency braking, stationary accidents, and hazardous road conditions like poor visibility or slippery surfaces.

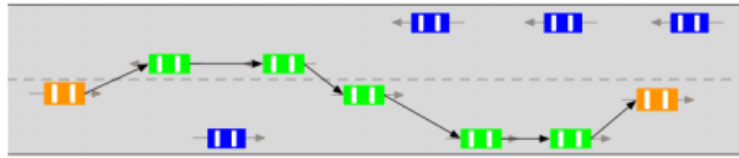
5. **Advertised Services** These messages are used to inform nearby vehicles about available services, such as parking information or entertainment downloads. They are typically broadcast, and vehicles can connect to these services based on the information provided.
- Objective: To announce available services to vehicles in the vicinity.
  - Key Characteristics:
    - Broadcast by service providers
    - Single-hop communication
    - Not time-sensitive

Examples include public transport announcements, parking availability, and media download services.

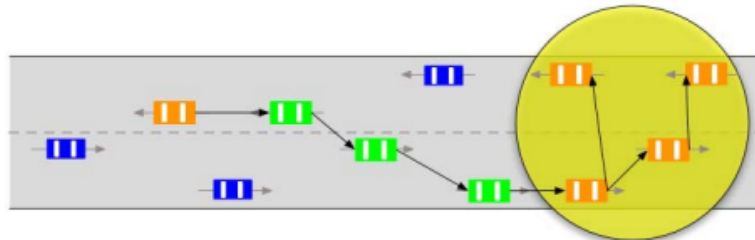
## 2.2 Geonetworking Protocol

The GeoNetworking (GN) protocol represents a specialised network-layer protocol [10], designed for use in mobile ad hoc communications within vehicular networks, with particular reference to VANETs (Vehicular Ad-Hoc Networks). The principal

objective of the protocol is to facilitate communication between vehicles (V2V) and between vehicles and infrastructure (V2I) by capitalising on the geographical locations of nodes for data transmission. The GN protocol has been specifically designed to address the dynamic and constantly shifting topologies that are inherent to VANETs, which result from the high mobility of vehicles. The protocol provides a connectionless and fully distributed mechanism that can accommodate rapid changes in the network topology by utilising the geographical positions of communicating nodes as a basis for routing decisions. The protocol supports a range of communication modes, including unicast, anycast and broadcast, based on the geographical location of the destination or the region to which the data should be disseminated as is illustrated in Figure 2.2. and 2.3 It is particularly suited to ITS that require timely data transmission for safety-critical and traffic efficiency applications.



**Figure 2.2:** GeoUnicast communication



**Figure 2.3:** GeoBroadcast communication

### 2.2.1 GeoNetworking Packet Structure

The structure of a GeoNetworking packet as can be seen in reference [11] is designed in such a way as to support geographical routing in an efficient manner, while allowing flexibility for different communication types. In Figure 2.4 is possible to see the below structure:

1. **Media Access Control (MAC) header** is responsible for identifying the source and destination of the data packet, as well as the type of data being

transmitted.

- **Function:** The management of channel access in the wireless medium is a key function.
- **Importance:** It guarantees the uninterrupted transmission of messages, which is of paramount importance in a dynamic vehicular environment.

## 2. Logical Link Control (LLC) Header

- **Function:** It identifies the specific protocol that is in use.
- In the context of geonetworking, the Ethernet Type field is set to 0x8947, which serves to indicate that the packet in question is a geonetworking message.

## 3. GeoNetworking Header: This constitutes the fundamental component of the packet, comprising fields that are indispensable for routing and forwarding data in accordance with geographical location. The header can be subdivided into the following sections:

- **Basic Header:**
  - **Version:** This field indicates the version of the protocol in use.
  - **The Time-to-Live (TTL) field** allows the user to define the duration of time that a packet is allowed to remain in the network. This parameter determines the maximum duration for which a packet may remain within the network.
  - The maximum number of hops that a packet may traverse is indicated by the Max Hops field. This parameter specifies the maximum number of hops that a packet is permitted to traverse before being discarded.

This guarantees that the packet is constrained in its scope and does not persist indefinitely.

- **Common Header:** The Common Header is a standardised set of data that is transmitted at the beginning of any packet of data.
  - Includes geographical routing data, such as destination coordinates and indications for which protocol layer should process the packet next.
  - It is of paramount importance to ensure that packet handling is consistent across different geographical regions, and this can only be achieved through standardisation.
- **Extended Header:**

- This is optional and used when the packet needs to support special communication types such as GeoUnicast, GeoBroadcast, or Topologically-Scoped Broadcast.
  - It allows the packet to adapt to different vehicular scenarios.
4. **Payload:** Contains the actual data to be transmitted, typically from higher-layer protocols. In VANETs, this often includes CAMs or DENMs.

<b>Access Layer Header</b>	<b>GeoNetworking Header</b>	<b>Payload (optional)</b>
----------------------------	-----------------------------	---------------------------

**Figure 2.4:** GeoNetworking Header

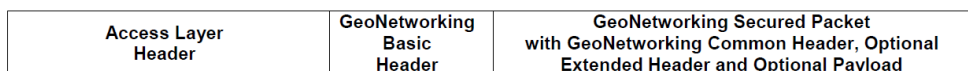
Once security features are enabled the packet structure evolves as illustrated in Figure 2.5. This results in the creation of a GeoNetworking Secured Packet, where the security header is inserted between the original GeoNetworking header and the payload. This transformation entails the following:

- Access Layer Header: The standard lower layer header.
- GeoNetworking Basic Header: Remains the same, handling geographic routing and addressing.
- GeoNetworking Secured Packet: This section contains the Common Header, optional Extended Header, and the newly added security components. These security fields ensure that data is protected via cryptographic means.

The security header encapsulates critical information such as:

- Digital signatures for verifying the authenticity of the sender.
- Certificates for validating the identity of the sender.
- Encryption-related fields for ensuring the confidentiality and integrity of the data.

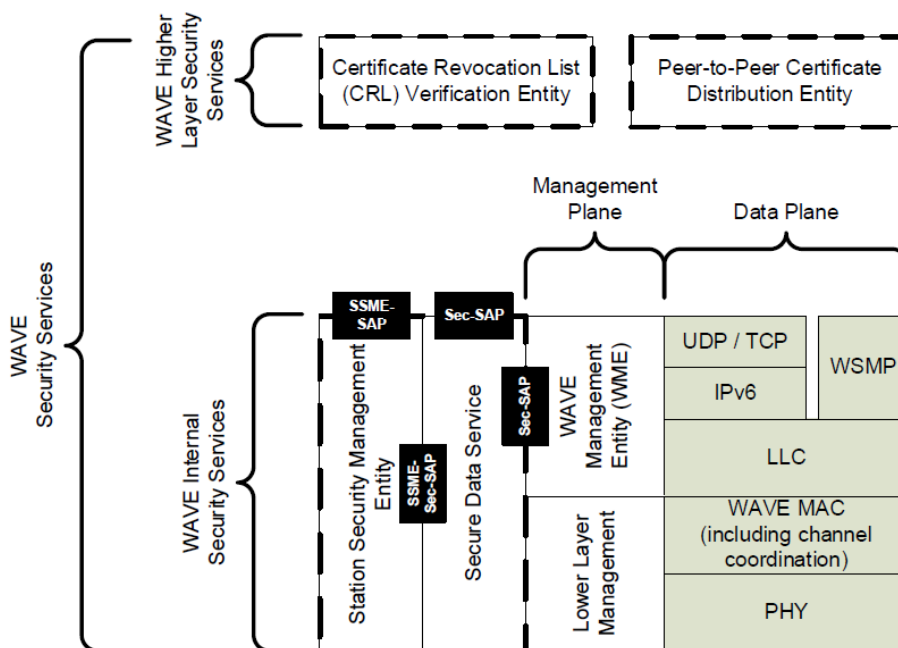
The insertion of the security components results in the GeoNetworking packet becoming a secured unit, protecting the data as it traverses the network without altering the core functionality of the routing headers. The security header thus augments the packet by adding cryptographic safeguards, ensuring that the message can be trusted while still adhering to the GeoNetworking protocol. This modification to the packet structure maintains the original GeoNetworking headers but adds layers of protection to address security threats.



**Figure 2.5:** GeoNetworking with security

## 2.2.2 Wireless Access in Vehicular Environments (WAVE)

In order to facilitate secure and reliable communication in vehicular networks, the Wireless Access in Vehicular Environments (WAVE) architectural framework is employed. WAVE provides a framework for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, thereby enabling rapid and low-latency data exchange between vehicles and road infrastructure. WAVE operates on both IP-based and non-IP-based protocols, including the WAVE Short Message Protocol (WSMP) for time-critical messages. It is constructed upon the IEEE 802.11p standard for wireless communication at the physical and medium access control (MAC) layers, thereby ensuring resilience in fast-moving environments such as highways.



**Figure 2.6:** WAVE protocol stack showing detail of WAVE Security Services

WAVE's security is defined in IEEE 1609.2 [12], which introduces a comprehensive set of security services to secure WAVE management and application messages. The security services layer provides critical functions such as message

signing, encryption, and validation, which ensure that messages are protected against tampering, eavesdropping, and replay attacks. A vital component of this security framework is the Security Services Protocol Data Unit (SPDU).

The internal security services offered by WAVE are as follows:

- **Secure Data Service (SDS):** It is a data security solution designed to safeguard sensitive information. The process of transforming unsecured protocol data units (PDUs) into security services protocol data units (SPDUs) for transfer between entities is a fundamental aspect of the security services framework. Additionally, the processing of SPDUs upon reception, including the transformation of SPDUs into unsecured PDUs, is a crucial element in maintaining the integrity and confidentiality of transmitted data. The additional data appended to a PDU upon its transformation into an SPDU is designated the security envelope. An entity that utilises the Secure Data Service is designated a Secure Data Exchange Entity (SDEE).
- **Security management:** Managing information about certificates.

WAVE Higher Layer Security Services are as follows:

- **Certificate revocation list (CRL) verification entity (CRLVE):** This entity validates incoming CRLs and passes the related revocation information to the SSME for storage.
- **Peer-to-peer certificate distribution (P2PCD) entity (P2PCDE):** This entity enables peer-to-peer certificate distribution.

### 2.2.3 Security services protocol data units (SPDUs)

The Security Services Protocol Data Unit (SPDU) [12] plays a pivotal role in ensuring the security of communications within the WAVE architecture. SPDUs encapsulate the essential cryptographic transformations, including signing, encryption, and certificate management. The structure of an SPDU is highly flexible, enabling it to convey both signed and encrypted data while maintaining compatibility with WAVE protocols. SPDUs are encoded using ASN.1, which guarantees the secure serialisation and deserialisation of the data for transmission across vehicular networks. The following is an overview of the principal SPDU employed in this work of thesis, as can be seen in Figure 2.7, with a particular emphasis on their respective functions:

#### 1. **Ieee1609Dot2Data**

- **protocolVersion:** Defines the version of the protocol being used (e.g., version 3 in CAM)

- **content**: Holds the actual payload, which can be signedData, unsecuredData, encryptedData, or certificate-related requests

## 2. **Ieee1609Dot2Content**

- **signedData**: Refers to data that has been cryptographically signed for authenticity
- **unsecuredData**: Data that has not undergone any security transformation (e.g., no encryption or signing)
- **encryptedData**: Data that has been encrypted to ensure confidentiality
- **signedCertificateRequest**: A request for a certificate that has been signed
- **signedX509CertificateRequest**: A signed request for an X.509 certificate

## 3. **SignedData**

- **hashId**: The hash identifier of the message or entity
- **tbsData (ToBeSignedData)**: Data that needs to be signed, containing the actual payload and header info
- **signer**: Identifies the entity or certificate signing the data
- **signature**: The digital signature ensuring the authenticity of the data

## 4. **ToBeSignedData**

- **payload**: The actual message or data being transmitted
- **headerInfo**: Contains metadata

## 5. **HeaderInfo**

- **psid**: Protocol Service Identifier, identifying the type of service or application
- **generationTime**: The time at which the message was generated

## 6. **CertificateBase**

- **version**: The version of the certificate
- **type**: The type of certificate being issued
- **issuer**: The entity that issued the certificate
- **toBeSigned**: Data structure that is to be signed within the certificate
- **signature**: The digital signature on the certificate

## 7. **ToBeSignedCertificate**



- `id`: The unique identifier of the certificate
- `crcaId`: The identifier of the Certificate Revocation Authority (CRA)
- `crlSeries`: Indicates the Certificate Revocation List (CRL) series
- `validityPeriod`: Defines the start and end of the certificate's validity period
- `appPermission`: Application permissions associated with the certificate
- `verifyKeyIndicator`: The indicator showing whether the key can be used for verification purposes

#### 8. **SignerIdentifier**

- `self`: Indicates that the signer is the entity itself
- `digest`: The cryptographic hash used for verification
- `certificate`: The certificate used by the signer

#### 9. **Signature**

- `rSig`: One part of the cryptographic signature
- `sSig`: The other part of the cryptographic signature

#### 10. **PsidSsp**

- `psid`: Protocol Service Identifier, describing the service
- `ssp` (bitmap-based): Security Service Profile, indicating the security services applicable to the message

#### 11. **VerificationKeyIndicator**

- `verificationKey`: The public key used to verify the signature
- `reconstructionValue`: A value used for key reconstruction

#### 12. **PublicVerificationKey**

- `ecdsaNistP256 / ecdsaNistP384`: Public keys for ECDSA using the NIST P-256 and P-384 curves
- `verifyKeyIndicator`: Indicates whether the key can be used for verification

#### 13. **EccP256CurvePoint**

- `x-only`: A single coordinate of the elliptic curve point
- `fill`: Padding for the field
- `compressed_y_0 / compressed_y_1`: The compressed form of the y-coordinate of the elliptic curve point
- `uncompressed (x,y)`: The uncompressed x and y coordinates

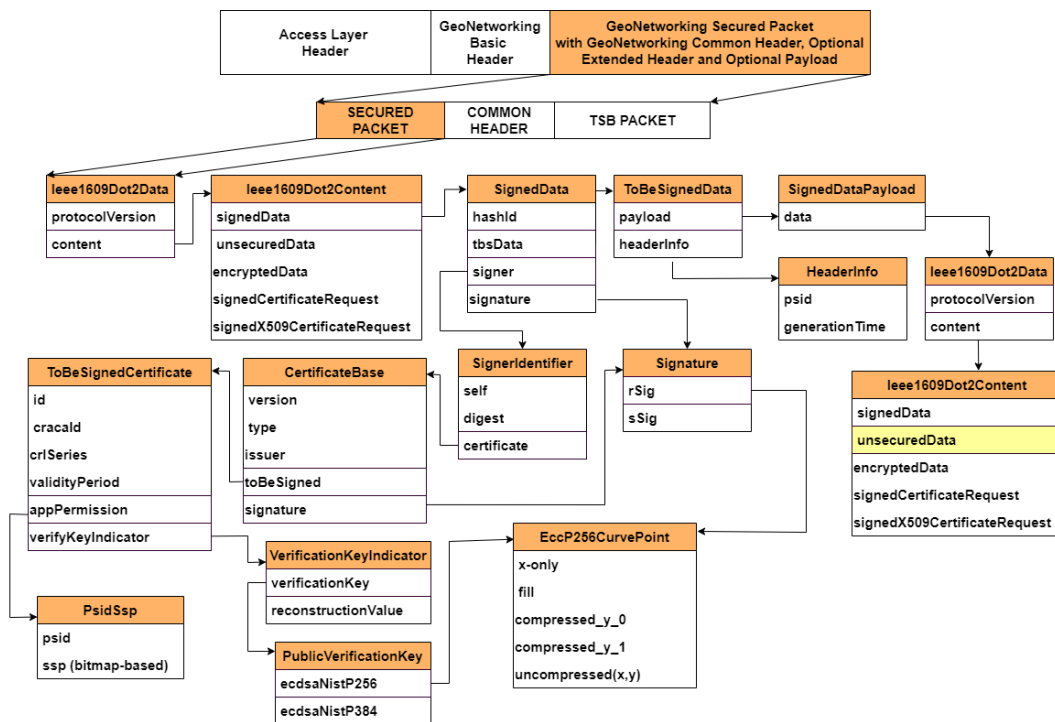


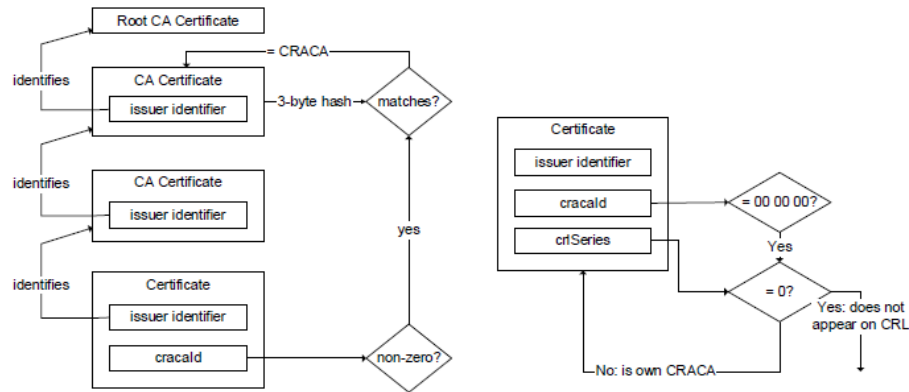
Figure 2.7: Security header SPDUs

## 2.2.4 Certificate Revocation Method

In the context of certificate management within vehicular networks, the *cracaId* and *crlSeries* fields are of critical importance in determining the manner and circumstances under which a certificate may be revoked. In accordance with the standard [12] and as is illustrated in Figure 2.8, the *cracaId* field is defined as a 3-byte octet string, serving to identify the Certificate Revocation Authority (CRA) and the Certificate Authority (CA) responsible for the certificate in question. In order to ascertain the pertinent CRACA certificate, the standard mandates the calculation of the hash of the certificate and the extraction of the low-order three bytes, designated as the HashedId3. If the three bytes in question match the *cracaId*, the certificate in question is linked to the corresponding CRACA.

In the event that the *cracaId* contains exclusively zeros, a special case is triggered. In such instances, a certificate with a *crlSeries* value of 0 indicates that it will not be revoked, either because the certificate has a very short validity period or because it employs an alternative revocation mechanism that is not in accordance with the standard. Conversely, if the *crlSeries* value is non-zero while the *cracaId* remains all zeros, this indicates that the certificate in question may still appear on a revocation list (CRL) that it signs itself.

In scenarios where the *cracaId* is non-zero but fails to match uniquely with a certificate in the full certificate chain, either by not being found or by matching multiple certificates, the certificate is deemed invalid. This revocation method ensures that each certificate can be properly associated with a CRACA and its corresponding revocation list, thereby maintaining the integrity and security of the system.



**Figure 2.8:** Logic flows used in determining the CRACA for a certificate

## 2.3 ITS Security Management System

As outlined in [9], The ITS Security Management System is founded upon a Public Key Infrastructure (PKI), which establishes a hierarchical structure for the administration, issuance, and authentication of digital certificates. This infrastructure guarantees the secure communication of Intelligent Transport Systems (ITS) by verifying the identity of the relevant entities, including vehicles, roadside units, and other components. The system comprises a number of key functional elements, including the Root Certification Authority (Root CA), the Enrolment Authority (EA), and the Authorisation Authority (AA).

A Public Key Infrastructure (PKI) is a system that enables secure communication through the use of encryption and digital signatures. The system comprises several key functional elements, including Certification Authorities (CAs), which issue digital certificates that associate a public key with the identity of an entity, whether that be a vehicle or device. Digital certificates serve to confirm the identity of the entities in question, while public/private key pairs are employed to secure communication through the use of encryption and digital signatures, thereby ensuring the integrity and confidentiality of the data exchanged. Moreover, PKI

systems utilise revocation lists, which permit the invalidation of compromised certificates.

The C-ITS Trust Model is based on the PKI structure but has been adapted for use in the context of Cooperative-ITS. This architectural framework facilitates the management of the lifecycle of trusted ITS stations, encompassing the issuance, administration, and revocation of certificates. The C-ITS Trust Model adheres to a hierarchical structure, as illustrated in Figure 2.9. The Root Certification Authority (Root CA) occupies the pinnacle of the structure, bearing the responsibility of issuing credentials to subordinate CAs, including the Enrolment Authority (EA) and Authorization Authority (AA). The Enrolment Authority is responsible for authenticating ITS stations and issuing enrolment certificates, which serve as a form of authentication and allow the stations to communicate securely within the ITS environment. The Authorisation Authority is charged with the issuance of authorisation tickets or pseudonym certificates, which enable stations to access specific services while maintaining privacy.

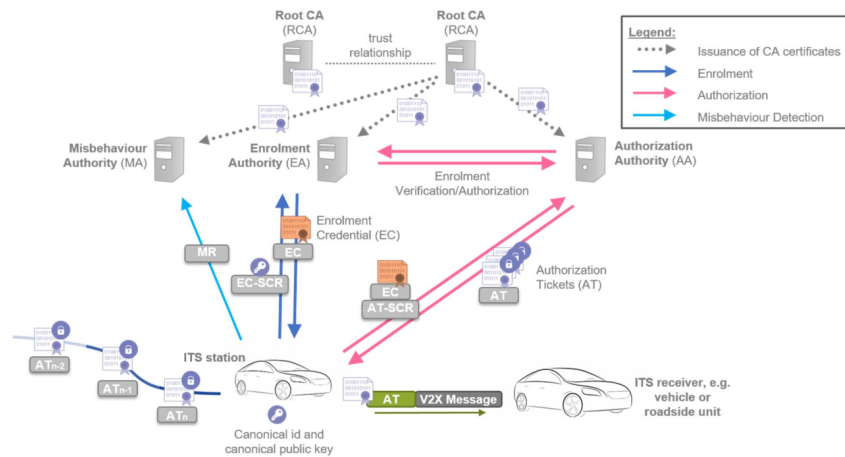
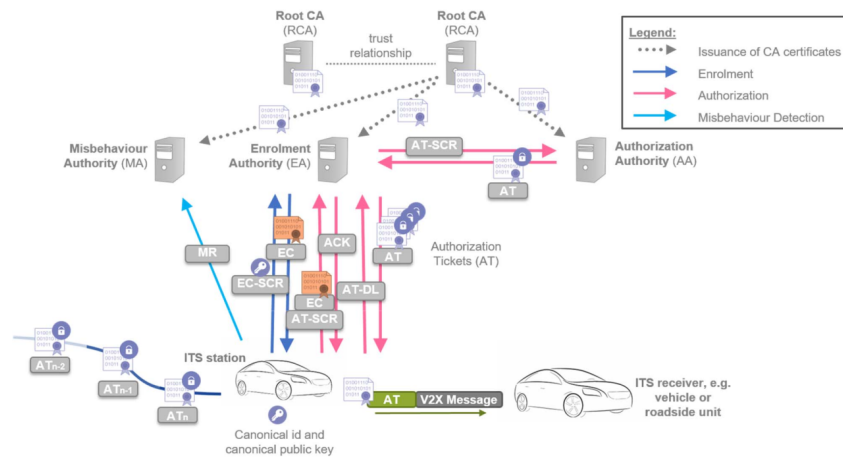


Figure 2.9: PKI architecture

### PKI architecture with butterfly AT provisioning

A significant distinction emerges when contrasting the conventional PKI model with the butterfly provisioning scheme illustrated in Figure 2.10. In the butterfly model, the Enrolment Authority also performs the function of a Registration Authority, thus enabling the batch issuance of certificates. This allows ITS stations to receive them in batches rather than individually. This results in a more efficient authorisation process, which is particularly advantageous when a large number of certificates need to be provisioned in a timely and cost-effective manner. The

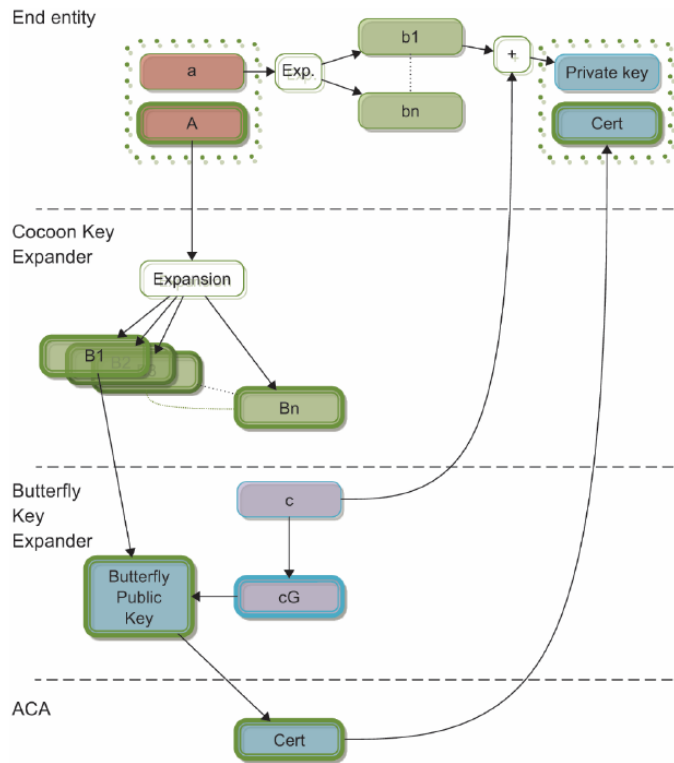
butterfly scheme offers a more scalable approach than the traditional PKI model, which involves a more sequential and individualised process. In this model, the stations interact with the Root CA, Enrolment Authority, and Authorisation Authority in a step-by-step manner. In this system, the Root CA serves as the ultimate trust anchor, providing the foundational trust by issuing credentials to the Enrolment Authority and Authorisation Authority. The Enrolment Authority is responsible for overseeing the entire enrolment process, verifying the identity of ITS stations and granting them access to the system through the issuance of enrolment certificates, while the Authorisation Authority issues authorisation tickets to the stations, which permit them to access specific services securely. Furthermore, a Trust List Manager is tasked with maintaining a list of trusted certificates within the system, while a Misbehaviour Authority is responsible for detecting and handling any misbehaving stations, potentially revoking their certificates if necessary.



**Figure 2.10:** PKI architecture with butterfly AT provisioning

The butterfly key provisioning scenario represents a more advanced and efficient approach to key management. The vehicle generates a single base key pair and applies a mathematical transformation, illustrated in Figure 2.11, to derive several variant keys. This transformation is known as the butterfly key expansion.

The butterfly key expansion algorithm described in [13] commences with the end entity generating a caterpillar key pair, comprising a private and a public key. The public key is expanded into cocoon keys using an expansion function, thereby ensuring that they are statistically uncorrelated. Subsequently, a butterfly key expander adds a random elliptic curve point to each cocoon key, thus creating a final butterfly public key. This key is employed in the generation of certificates. Only the end entity, in possession of its caterpillar private key, is able to derive the corresponding butterfly private key, thereby guaranteeing the confidentiality and



**Figure 2.11:** Butterfly key expansion

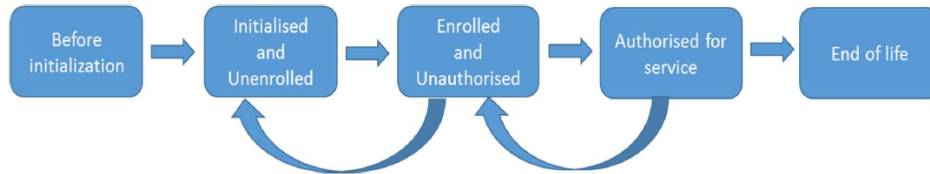
security of the process.

### 2.3.1 ITS station security lifecycle

The ITS station security lifecycle, as outlined in [14] and shown in Figure 2.12, encompasses a series of stages for the administration of security credentials for ITS stations (vehicles or roadside units). The aforementioned lifecycle guarantees the security of communication through the generation, utilisation, renewal and revocation of digital certificates. The process typically commences with the initialization phase, during which an ITS station receives a canonical key pair and is assigned a canonical identity. Subsequently, the station initiates the enrolment phase, wherein it requests an enrolment certificate from the Enrolment Authority (EA). The certificate is of paramount importance for the authentication of the station within the system.

Subsequently, during the authorisation phase, the station utilises the aforementioned enrolment certificate to request authorisation tickets from the Authorisation Authority (AA). These tickets, which are pseudonym certificates, are used for

secure and anonymous communication within the system, thereby ensuring privacy. The lifecycle also includes regular certificate renewal in order to maintain trust and certificate revocation, which occurs if the station misbehaves or the credentials are compromised. Throughout its lifecycle, an ITS station undergoes multiple checks in order to ensure the integrity and validity of its certificates.



**Figure 2.12:** ITS Station Security Life Cycle

## 2.3.2 Cryptographic operations

### Encryption algorithm - ECDH

In [12] is addressed the encryption of messages from the sender to the receiver in the context of Intelligent Transport Systems (ITS) communication. The encryption algorithm most commonly utilised is based on Elliptic Curve Cryptography (ECC), which provides a secure and efficient method of ensuring data confidentiality.

The process entails the generation of a shared encryption key between the sender and receiver through the utilisation of the elliptic curve Diffie-Hellman (ECDH) method. This key is employed for the encryption of the message payload. ECC is preferred due to its robust security at smaller key sizes in comparison to traditional encryption algorithms, thereby ensuring the secure transmission of ITS messages without undue computational overhead.

The encrypted message is then transmitted, ensuring that only the intended recipient, in possession of the corresponding private key, is able to decrypt and access the original data.

### Signature algorithm - ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA), as detailed in reference [14], provides two digital signature algorithms. The standard makes a particular point of emphasising the use of ECDSA based on the NIST P-256 and NIST P-384 elliptic curves for the purposes of secure message signing and verification. Furthermore, the standard makes reference to the potential utilisation of curves such as brainpoolP256r1 and brainpoolP384r1, with the objective of achieving broader compatibility.

The NIST P-256 curve is typically employed for the signing of PDUs (Protocol Data Units) and certificates associated with message exchange. In contrast, NIST P-384 or brainpool P-384 are reserved for higher-security cases, such as root CA (Certificate Authority) certificates and Trust List Manager (TLM) certificates. The use of a certificate encoded with these curves and algorithms ensures the trustworthiness and authenticity of the transmitted data.

Key Elements for Hashing with ECDSA:

- The standard permits the utilisation of either SHA-256 for NIST P-256 curves or SHA-384 for NIST P-384 curves to generate hashes that will serve as the input for the signature algorithm.
- The data to be signed is hashed and combined with the Signer Identifier Input to create a final hash for the signing process. The Signer Identifier Input may be either a certificate if verification is required or a "self-signed" identifier (an empty string).

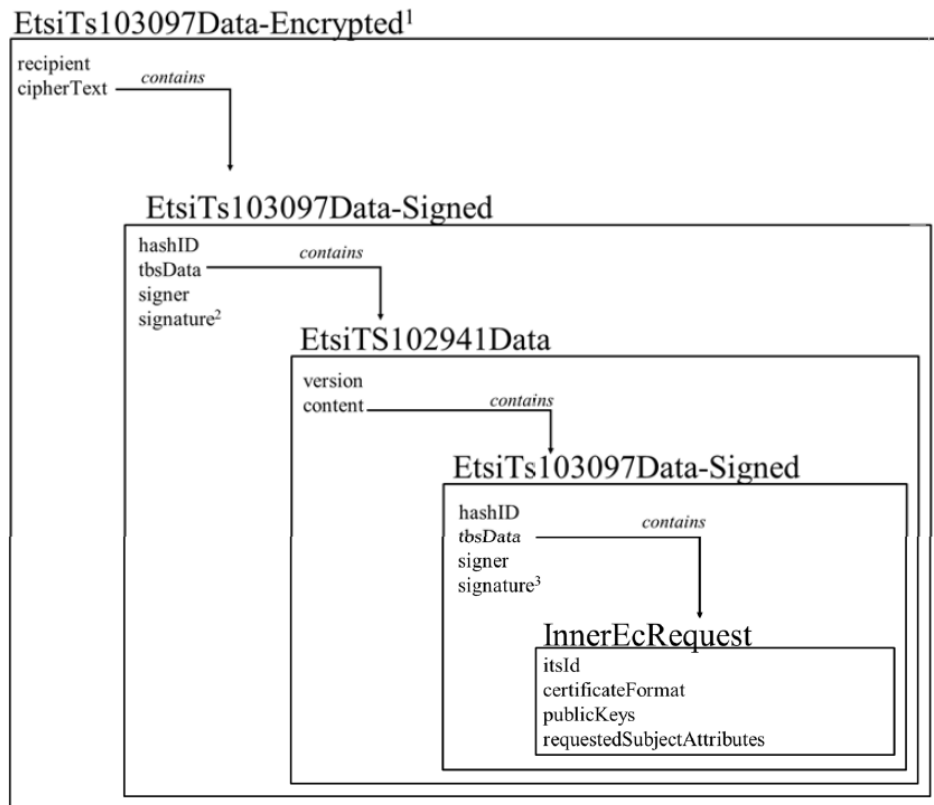
This signature process provides integrity and authenticity for data packets, thereby ensuring secure communication between systems. The encoding of these hashes adheres to the detailed rules specified in the standard, thus allowing them to be used in conjunction with GeoNetworking and other ITS protocols.

### 2.3.3 Enrolment Management

The enrolment process is responsible for granting an ITS station an enrolment certificate, which is required for subsequent authorization requests. This certificate essentially acts as proof of the ITS station's identity within the system. The key steps involved are:

1. Request for Enrolment Certificate (illustrated in Figure 2.13):
  - The ITS station transmits an enrollment request to the EA.
  - The request is encrypted using a public key provided by the EA and includes a newly generated key pair for the ITS station.
  - The canonical identity, which serves as a globally unique identifier for the ITS-S, is linked to the station during this process.
  - The ITS station authenticates the request by appending its private key to the data, thereby providing proof of possession, and transmits it to the EA.
2. Enrolment Credential Issuance:





**Figure 2.13:** Enrolment Request structure

- The EA is responsible for verifying the authenticity and correctness of the request. If the request is deemed valid, the EA issues an enrolment certificate, which includes the station's public key, and sends it back to the ITS station in an encrypted response.
- This certificate serves as a form of identification for future communication with other entities, particularly for the purpose of obtaining authorisation tickets from the AA.

### 3. Key Components:

- Canonical key pair: It is a private-public key pair generated by the ITS station. The private key is kept secure, while the public key is shared as part of the enrolment request.
- Enrolment Credential (EC): It is a certificate that associates a station's identity with its public key, which is issued by the EA. As illustrated in Figure 2.14, the certificate can be retrieved from the EC response.

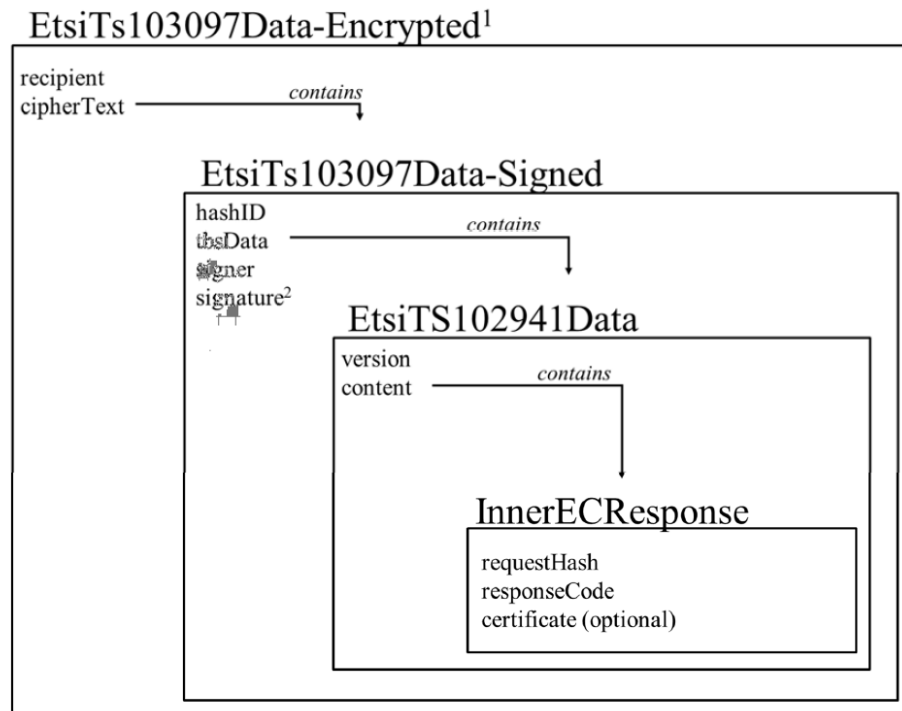


Figure 2.14: Enrolment Response structure

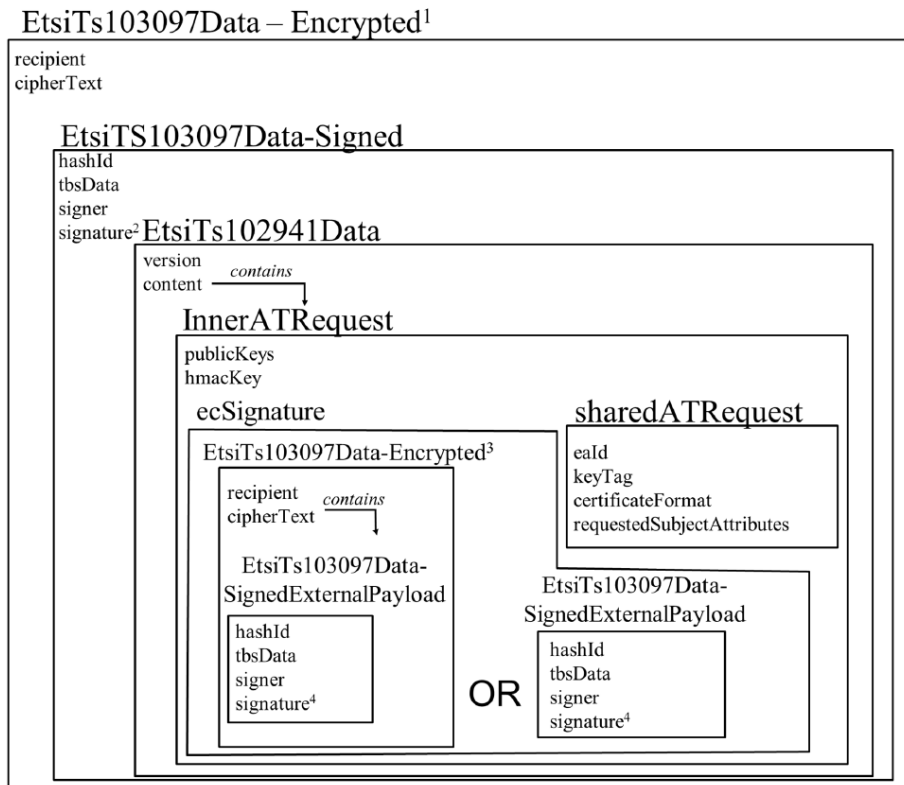
### 2.3.4 Authorization Management

The process of authorisation involves the acquisition of pseudonym certificates, otherwise known as authorisation tickets, which permit communication in a secure manner without disclosing the identity of the station in question. These pseudonym certificates serve to uphold the integrity and confidentiality of the communication process. The procedure is outlined below:

1. Authorization Request:

- The ITS station transmits an Authorisation Request to the AA, which includes the enrolment certificate. The request is encrypted with the algorithm described in reference 2.3.2, and its complete structure is illustrated in Figure 2.15.
- The request is typically digitally signed using the private key corresponding to the public key in the enrolment certificate, thereby proving possession of the private key.

2. Authorization Ticket Issuance:



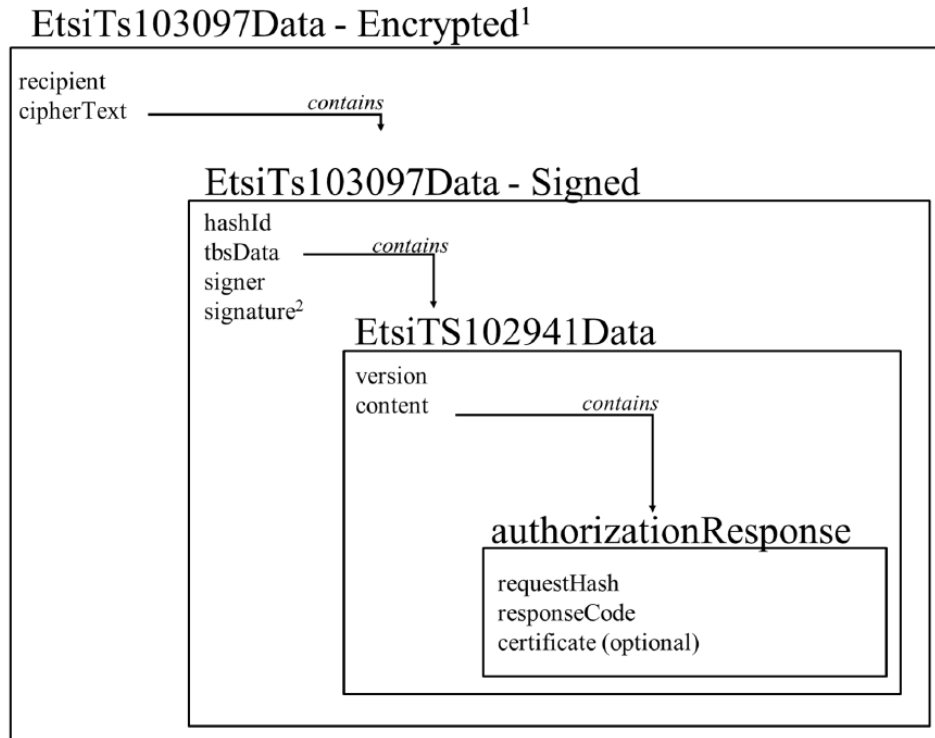
**Figure 2.15:** Authorization Request structure

- Upon receipt of the request, the AA initiates the validation process, which, if successful, culminates in the issuance of an authorization ticket (AT).
- The authorisation ticket enables the station to send digitally signed and encrypted messages to other ITS stations while maintaining the confidentiality of its communications by using pseudonyms in place of its canonical identity.

### 3. Key Components:

- **Authorization Ticket (AT):** It is a pseudonym certificate that enables ITS stations to communicate securely without disclosing their canonical identity. It is a crucial component for ensuring privacy protection within the system. The AA response, illustrated in Figure 2.16, allows for the recovery of the pseudonym certificate.
- **Public/Private key pair:** It is generated by the ITS station for each enrolment. This key pair is then associated with the relevant authorisation

ticket. The private key is used to sign future messages in a secure manner, while the public key is included in the ticket.



**Figure 2.16:** Authorization Response structure

## 2.4 Security profiles and certificate formats

Security profiles, as defined in [15], delineate a set of rules and configurations designed to ensure the security of communication within intelligent transport systems (ITS). These profiles provide a flexible yet standardised approach to ensuring the protection of messages exchanged between vehicles, between vehicles and infrastructure, and other V2X communications against a range of potential security threats. Each security profile delineates the requisite specifications for message authentication, integrity, and confidentiality, as well as the implementation of these requirements through the use of cryptographic techniques, including digital signatures and encryption.

A crucial aspect of these security profiles is their function in guaranteeing interoperability between diverse devices and systems, while enabling flexibility in the level of security applied contingent on the context of communication. For

instance, messages with elevated security risks or that necessitate confidentiality may utilise more sophisticated security mechanisms, whereas lower-risk communications may have more streamlined requirements to enhance performance.

### 2.4.1 CAM profile

The security profile for Cooperative Awareness Messages (CAMs) has been developed with the objective of safeguarding one of the most elementary types of messages exchanged in V2X communication. CAMs are periodic broadcast messages that provide real-time information about a vehicle's status, including its position, speed, direction, and other relevant details. It is therefore evident that these messages are of the utmost importance for enabling vehicles to be aware of each other and to coordinate their movements in a safe and efficient manner.

In view of the high frequency of CAM transmission, it is of paramount importance to achieve an optimal balance between performance and security in the security profile. The profile stipulates that each CAM must be digitally signed in order to guarantee its authenticity and integrity. This enables the recipient to verify that the message has been transmitted by the intended source and has not been altered during transmission. It should be noted, however, that confidentiality is not required and thus the messages are not encrypted.

Here are the significant fields and their roles in securing CAMs:

#### 1. Message Fields:

- **Generation Time:** The timestamp indicates the point in time at which the CAM was created. This field is useful for verifying the freshness of the message, which is crucial for preventing replay attacks.
- **Payload Data:** The vehicle's status information is also included in the data set, which comprises details such as the vehicle's position, speed, heading and so forth. This constitutes the fundamental data set of a CAM.
- **Security Header:** The field in which security-related metadata is stored, including details such as the cryptographic algorithms used.

#### 2. Security Fields:

- **Signature:** CAMs must be signed to ensure authenticity and integrity. The digital signature is applied to the payload data and other relevant fields using a private key.
- **Signer Info:** Contains information regarding the certificate or pseudonym utilized to authenticate the CAM. This information typically includes a reference to the pseudonym certificate.

- Pseudonym Certificate: The use of pseudonym certificates by CAMs serves to guarantee the confidentiality of the vehicle's identity while simultaneously confirming the veracity of the transmitted data. The certificate in question establishes a link between the public key and a pseudonym, thereby safeguarding the privacy of the vehicle's owner.

3. Security Mechanisms:

- HashedId8: This is a hash-based identifier (8 bytes) derived from the pseudonym certificate to ensure that the vehicle's real identity remains private but still allows for message authentication.
- Replay Protection: The reuse of old messages in replay attacks can be avoided by utilising time-based mechanisms, such as the Generation Time field.
- In order to enhance privacy, the vehicle is programmed to change its pseudonym certificate at regular intervals. This entails the use of a new certificate after a specified time period or after a predefined number of CAM messages.

# Chapter 3

## Formal Verification

### 3.1 Introduction

Formal verification is a process that employs mathematical techniques to demonstrate, or otherwise invalidate, the correctness of a system with respect to a specific set of properties or specifications. In the context of cybersecurity and critical infrastructure, formal verification plays a pivotal role in providing robust assurances that a system adheres to its intended behaviour. In contrast to conventional testing techniques, which assess system behaviour against a limited set of scenarios, formal verification employs an exhaustive evaluation of all potential states of a system, thereby guaranteeing that it fulfils its security and safety requirements in all eventualities.

In complex, high-stakes systems such as automotive safety, aerospace, and critical infrastructure, where any security vulnerability or system failure could have severe consequences, formal verification is of particular importance. The principal approaches to formal verification encompass model checking and theorem proving, both of which offer distinct advantages contingent on the complexity and character of the system undergoing analysis [16].

#### 3.1.1 Model checking

Model checking is an automated formal verification method that is employed to ascertain whether a finite-state model of a system satisfies the requisite properties. The system is represented as a state machine, and the properties to be verified are typically expressed in formal logics, such as temporal logic.

Model checkers exhaustively explore the state space of the model to either confirm that the properties hold or identify instances where the system does not meet the required specification. The automation and immediate feedback provided by model checking make it a widely used method, particularly in the early stages

of system design, where it can prevent costly revisions by identifying and fixing design flaws at an early stage.

### 3.1.2 Theorem Proving

Another approach to formal verification is theorem proving, which involves the creation of formal mathematical proofs to demonstrate that a system meets its specified criteria. In contrast to model checking, which automatically verifies all potential states, theorem proving necessitates human involvement to direct the process of establishing or refuting specific properties. The process employs formal logic and proof assistants, which facilitate automation of certain proof-related tasks but necessitate greater manual input than is required in the case of model checking.

The use of theorem proving is frequently employed in the context of systems where the feasibility of model checking is limited by the extent or intricacy of the state space. It is particularly advantageous in the context of establishing the correctness of algorithms, protocols, and systems that exhibit infinite or highly complex behaviours, which cannot be readily represented through the use of finite-state machines. While it necessitates a greater investment of effort and expertise, theorem proving offers a superior level of assurance when applied to critical security and safety properties.

## 3.2 ProVerif

ProVerif is an automated cryptographic protocol verifier, as outlined in reference [17], predominantly employed for the assessment of cryptographic protocol security. The software is designed to handle both reachability properties, such as ensuring that secret data cannot be accessed by an attacker, and observational equivalence, which is used to prove that two processes behave indistinguishably from the perspective of an attacker. ProVerif is particularly efficacious for the analysis of properties such as secrecy, authentication, privacy, and traceability within the context of cryptographic protocols.

ProVerif operates by encoding protocols in a variant of the applied pi calculus, which is a formal language for describing concurrent systems. The tool translates these protocol descriptions into Horn clauses and employs resolution techniques to demonstrate the validity of the desired properties. One of its most notable features is its capacity to handle an unbounded number of sessions and an unbounded message space, which renders it highly applicable to real-world cryptographic protocols. Furthermore, it is capable of supporting a multitude of cryptographic primitives, including symmetric and asymmetric encryption, digital signatures, hash functions, and zero-knowledge proofs.



In the context of formal verification methodologies, ProVerif is primarily regarded as a model checker, as it automatically validates the properties of a protocol by analysing its state space. However, in contrast to traditional model checkers that explicitly explore all potential states, ProVerif employs logical inference through Horn clauses to circumvent the construction of an exhaustive state space. This methodology enables it to accommodate infinite state spaces and unbounded protocol sessions, thereby blurring the distinction between model checking and theorem proving. While it retains the automation characteristic of model checking, its utilisation of logical proofs for specific properties bears resemblance to aspects of theorem proving.

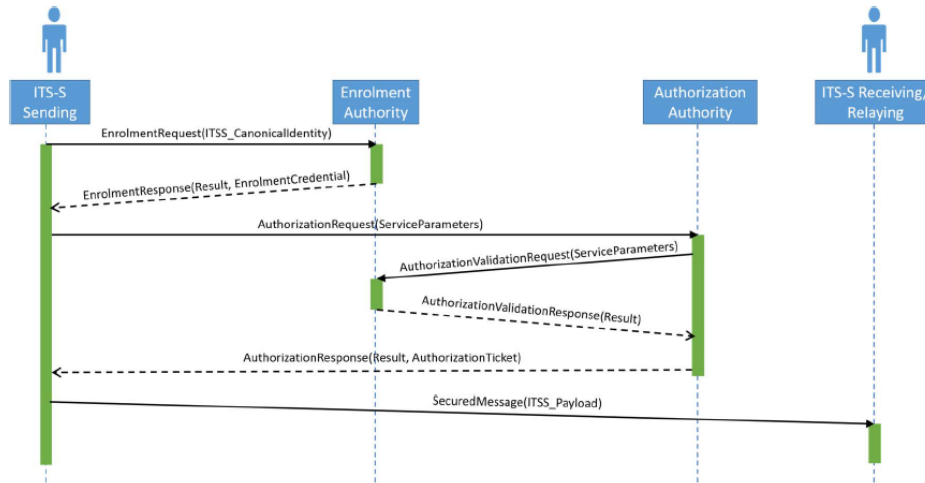
Furthermore, in instances where ProVerif is unable to verify a security property, it is capable of reconstructing attack traces, thereby offering invaluable insight into potential protocol vulnerabilities. This combination of automated verification, flexibility in handling large or infinite state spaces, and the ability to generate attack traces makes ProVerif an extremely powerful tool for cryptographic protocol analysis.

### 3.2.1 PKI verification

This section presents the application of ProVerif for the modelling and verification of the security of the Public Key Infrastructure (PKI) within the V2X ecosystem. The objective is to guarantee that the interactions between key entities, as delineated in 2.3, comply with essential security properties, including secrecy, authentication, reachability, and non-interference. ProVerif, with its capacity to process cryptographic primitives and verify security protocols, represents an optimal tool for analysing the PKI process that governs the secure communication between vehicles and authorities in the V2X network.

The verification model is centred on the interactions between three principal entities as shown in Figure 3.1: the Intelligent Transportation System Station (ITS-S), the Enrolment Authority (EA) and the Authorisation Authority (AA). The initial step is for the ITS-S to register with the EA in order to obtain an Enrolment Certificate (EC). Once this has been completed, the ITS-S proceeds to the authorisation phase, in which it contacts the AA to request an Authorisation Ticket (AT). These certificates are of great importance in ensuring secure and authenticated communication within the V2X network, as they permit vehicles to prove their identity and legitimacy in order to communicate within the system.

The PKI process is founded on a system of public and private keys, with cryptographic operations such as encryption, decryption, and digital signatures ensuring the integrity and confidentiality of communication at each stage. In particular, the ITS-S and the relevant authorities (EA and AA) exchange messages that are encrypted and digitally signed, thereby ensuring that only authorised



**Figure 3.1:** Sequence to achieve signed message transfer between ITS-Ss

entities are able to access or manipulate the data. ProVerif is employed for the purpose of verifying that each of these steps maintains the desired security guarantees.

In order to verify the PKI model, ProVerif first encodes the interactions between the ITS-S, EA, and AA into the applied pi-calculus. This formal description permits the utilisation of Horn clauses to investigate the protocol’s behaviour across an infinite number of potential communication sessions. The protocol’s critical operations, including the ITS-S sending an EC request, the EA responding with a signed certificate, and the subsequent AT request to the AA, are encoded as processes in the applied pi-calculus. Each step involves the verification of cryptographic operations, including the authentication of entities through signature verification and the assurance of confidentiality through message decryption.

### 3.3 Verification Results

ProVerif then checks whether the model satisfies four key security properties:

- **Reachability:** In the context of security protocols, reachability refers to the ability of a particular state or action within the system to be "reached" or executed under specific conditions. To illustrate, a state may represent the access of secret or privileged information. If an attacker is able to "reach" such a state, it indicates that the protocol is vulnerable. It checks if an unauthorized party can access certain sensitive actions or data. The goal is to ensure that attackers cannot reach a state where they could compromise the

system's integrity, such as accessing private data or performing actions they should not be allowed to. It focuses on proving that the system never enters an insecure state.

```
Query not event(endAA) is false.
Query not event(endEA) is false.
Query not event(endITS_S) is false.
Query not event(endITS_R) is false.
```

**Figure 3.2:** Reachability property result

Reachability has been fully respected. This means that the analysis confirms that no attacker can "reach" sensitive parts of the system, ensuring that restricted terms or states are not exposed to unauthorized entities.

In ProVerif, this property is verified by checking whether the system enters unsafe states. The fact that reachability returns false, as shown in Figure 3.2, means that the system does not allow attackers to access any states or terms that would compromise security, indicating that the protocol is robust in terms of protecting against unauthorized access. The protocol can successfully complete its intended operations, such as the issuance of certificates and tickets, without encountering any deadlocks or unreachable states. In the verification process, ProVerif checks that each step of the enrollment and authorization phases can be fully executed without errors or interruptions, and that the ITS-S always receives its certificates as expected.

- **Secrecy:** It is a fundamental property of cryptographic protocols and its objective is to guarantee the confidentiality of sensitive information, such as cryptographic keys or private data, from potential exploitation by an adversary. Secrecy provides assurance that, even if an adversary is able to observe certain aspects of the communication, they should not be able to discern the protected information. This is a crucial aspect in ensuring the privacy of sensitive communications.

In the PKI model, ProVerif verifies that all secret data, including the private keys and session-specific information exchanged between the ITS-S, EA, and AA, remains confidential, as illustrated in Figure 3.3. The results confirm that there are no leaks of secret information, ensuring robust confidentiality.

- **Authentication:** The process of authentication represents a fundamental aspect of security protocols, serving to guarantee the veracity of the entities

```

Query not attacker(k1[l4 = v,l3 = v_1,l2 = v_2,l1 = v_3]) is true.
Query not attacker(k2[response = v,l4 = v_1,l3 = v_2,l2 = v_3,l1 = v_4]) is true.
Query not attacker(k3_1[req2 = v,req_1 = v_1,l4 = v_2,l3 = v_3,l2 = v_4,l1 = v_5]) is true.
Query not attacker(k4[response2 = v,response = v_1,l4 = v_2,l3 = v_3,l2 = v_4,l1 = v_5]) is true.
Query not attacker(EC_certificate_1[req_1 = v,l4 = v_1,l3 = v_2,l2 = v_3,l1 = v_4]) is true.
Query not attacker(AT_certificates_1[req = v,l4 = v_1,l3 = v_2,l2 = v_3,l1 = v_4]) is true.

```

**Figure 3.3:** Secrecy property result

engaged in communication by confirming their identity. It prevents impersonation attacks by verifying the legitimacy of the communicating parties. This is exemplified by the interaction between an ITS-S (Intelligent Transport System Station) and an Enrollment Authority (EA) or Authorization Authority (AA). This verification process is of paramount importance in order to guarantee that messages and actions originate from trusted and legitimate sources. With regard to your protocol, correspondence assertions are used to formalise the aforementioned relationships between events, thereby ensuring that particular actions, such as the receipt of a message, can only be carried out if the corresponding sending action was previously executed by a verified entity. Two types of correspondence assertions exist: injective and non-injective:

- The non-injective correspondence has been validated in both the standard AT provisioning scheme shown in Figure 3.4 and the Butterfly AT provisioning scheme illustrated in 3.5. This signifies that ProVerif corroborates the comprehensive sequence of events, ensuring that each action is aligned with a valid preceding one, thus substantiating the general authenticity of the process.
- In contrast, injective correspondence is not validated. Injective correspondence is more robust as it ensures the exclusivity of event pairing – for instance, if a message is received, it must correspond to a unique message that was sent. This exclusivity is vital to avert replay attacks, where an adversary reuses a previously captured message to deceive the system. The failure to verify the injective correspondence in this protocol is not the result of an oversight, but rather a limitation in ProVerif’s simulation capabilities. The protocol employs the use of nonces, which are random values utilized to ensure the freshness of messages, to prevent replay attacks, a common and effective defense mechanism. The issue arises from the inability of ProVerif to simulate whether the EA or AA (enrollment or authorization authorities) properly store, retrieve, and check nonces. In practice, these authorities are expected to maintain

a record of used nonces to prevent their reuse. However, ProVerif is unable to model this stateful behaviour, which introduces uncertainty. Consequently, while nonces are implemented in the protocol to avoid replay attacks, the inability to simulate the nonce tracking mechanism in ProVerif makes it impossible to formally verify injective correspondence.

```

Query event(EAend_EC(x,y,z)) ==> event(ITS_Sbegin_EC(x,y,z)) is true.
Query event(AAend_AT(x,y,z,w)) ==> event(ITS_Sbegin_AT(x,y,z,w)) is true.
Query event(EAend_validation(x,y,z)) ==> event(AAbegin_validation(x,y,z)) is true.
Query event(ITS_Rend_mess(x,y)) ==> event(ITS_Sbegin_mess(x,y)) is true.

```

**Figure 3.4:** Authentication property result - non-injective correspondence

```

Query event(EAend_EC(x,y,z)) ==> event(ITS_Sbegin_EC(x,y,z)) is true.
Query event(EAe_ButtReq(x,y,z,w)) ==> event(ITS_Sb_ButtReq(x,y,z,w)) is true.
Query event(AAe_ButtCertReq(x,y,z,w)) ==> event(EAb_ButtCertReq(x,y,z,w)) is true.
Query event(EAe_ATDownload(x,y,z,w)) ==> event(ITS_Sb_ATDownload(x,y,z,w)) is true.
Query event(ITS_Rend_mess(x,y)) ==> event(ITS_Sbegin_mess(x,y)) is true.

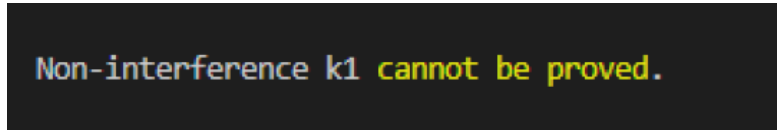
```

**Figure 3.5:** Authentication property result with Butterfly AT provisioning - non-injective correspondence

In conclusion, while the formal verification of injective correspondence was not possible due to the limitations in ProVerif’s ability to simulate nonce tracking by the authorities (EA and AA), the protocol still achieves mutual authentication between the ITS-S and the authorities. This guarantees that both parties can verify one another’s identities with a high degree of reliability, thereby preventing impersonation attacks. The absence of injective correspondence gives rise to potential concerns about replay attacks. However, the use of nonces within the protocol mitigates this risk by ensuring the freshness of messages in practice. Consequently, despite the formal limitation, the system offers robust guarantees of authentication.

- **Non-Interference:** The principle of non-interference is designed to prevent any observable changes in the system that could be exploited by an attacker to infer private information. This is particularly relevant in the context of sensitive data or operations, where the potential for such inferences to be made

is heightened. This implies that the protocol should exhibit the same behaviour irrespective of whether sensitive actions (such as modifying a password or updating a certificate) are carried out, from the perspective of an unauthorised observer. Non-interference precludes attackers from discerning alterations to sensitive data through observation of system behaviour. Even if an attacker is able to monitor the system, they should not be able to ascertain that a sensitive action, such as a password change or certificate update, has occurred merely by analysing response times or message patterns. The objective is to eradicate indirect information leakage through observable changes.



```
Non-interference k1 cannot be proved.
```

**Figure 3.6:** Non-interference property result

It was not possible to prove non-interference; for instance, Figure 3.6 illustrates the scenario of a shared key used for a request. This indicates the potential for an attacker to infer specific sensitive actions (e.g., certificate updates, password changes) through observation of system behaviour. Despite the lack of direct access to sensitive data (due to the maintenance of secrecy and reachability), the attacker may still be able to ascertain the occurrence of certain events through subtle system alterations, such as timing or message patterns. The inability to prove non-interference suggests that the protocol may potentially leak information indirectly, even if the core sensitive data remains secure.

Through these analyses, ProVerif not only confirms the soundness of the cryptographic operations involved but also provides assurances that the PKI implementation within the V2X framework is secure and resilient against various types of attacks.

# Chapter 4

## ms-van3t

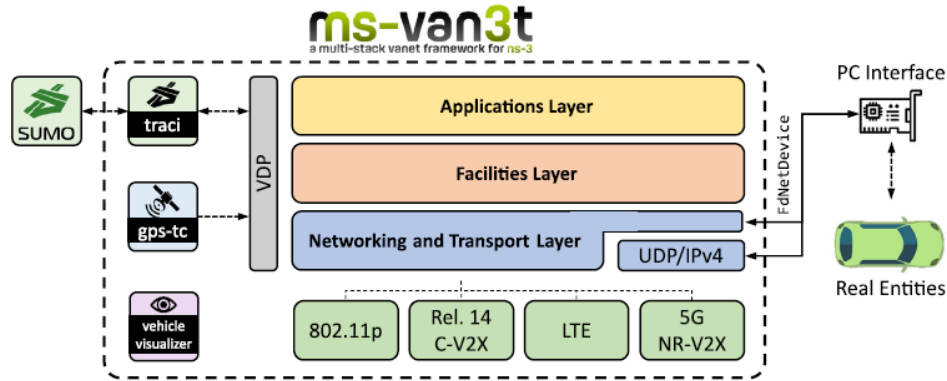
### 4.1 Introduction

The ms-van3t framework represents a comprehensive open-source platform for the simulation and validation of vehicle-to-everything (V2X) communications, as described in [18]. The ms-van3t framework is an integrated environment for modelling both the movement of vehicles and their communication networks. It is built upon the powerful ns-3 network simulator and SUMO (Simulation of Urban MObility) for vehicle mobility simulation. The framework supports various V2X communication technologies, such as IEEE 802.11p, LTE-V2X, and 5G NR-V2X, making it highly versatile for testing different vehicular communication protocols.

A significant attribute of ms-van3t is its capacity to integrate the comprehensive ETSI ITS-G5 protocol stack, encompassing a multitude of communication layers. These layers can be classified into the following categories:

- The Access Layer, responsible for physical and MAC layer functionalities, supporting 802.11p and C-V2X.
- The Network and Transport Layer, which incorporates protocols like GeoNetworking described in 2.2.
- The Facilities Layer, which handles the management of messages like Cooperative Awareness Messages (CAM) and Decentralized Environmental Notification Messages (DENM) illustrated in section 2.1.1.

This comprehensive architectural framework facilitates realistic simulations of V2V and V2I communications, enabling a diverse range of applications such as traffic management, collision avoidance, and emergency response. By integrating ms-van3t with SUMO, it ensures that vehicle movement patterns accurately reflect



**Figure 4.1:** Full architecture of ms-van3t

real-world urban and highway conditions, thereby providing a more authentic simulation environment.

One of the main strengths of ms-van3t is its support for multiple communication stacks, enabling users to simulate different communication protocols within the same environment. This makes it adaptable for evaluating different V2X scenarios, such as multi-hop communication, centralized and distributed applications, and hybrid simulation setups. In addition, ms-van3t allows Hardware-in-the-Loop (HIL) testing, which enables real-world hardware to interface with the simulated environment for more extensive evaluation.

The advancement of the security header, particularly the incorporation of digital signatures, represents a pivotal stride in the fortification of V2X communications. Digital signatures are of paramount importance in guaranteeing the veracity and integrity of messages exchanged between vehicles and infrastructure. The cryptographic verification of the sender’s identity provided by digital signatures offers a robust mechanism to prevent impersonation attacks, such as the Sybil attack that was described in detail in section 1.2.2. The authentication process ensures that each message originates from a legitimate and authorised source, making it exceedingly difficult for an attacker to forge multiple identities. Furthermore, the digital signature guarantees that the message has not been tampered with during transit, thereby enhancing the reliability of the communication network. This thesis, through the ms-van3t framework, will test and validate these mechanisms to demonstrate their efficacy in mitigating security threats while maintaining high performance in vehicular communication environments.



## 4.2 Security Header

The security header structure, which plays a vital role in ensuring secure V2X communications, was previously introduced and defined in section 2.2.1. This section will focus on the technical steps taken to implement and integrate the security header within the *ms-van3t* framework. These steps included the following: the ETSI ASN.1 files were compiled into usable C code; the CAM profile structure was populated and tested; and a digital signature mechanism was developed for message authentication and integrity.

### 4.2.1 ASN.1 Compilation and Integration into *ms-van3t*

The development of the security header started with the compilation of ASN.1 (Abstract Syntax Notation One) files in accordance with the specifications set forth in the ETSI standards. ASN.1 offers a formal and precise methodology for the description of the structure of data exchanged in V2X communications, thereby ensuring compliance with international standards and the capacity for interoperability between different systems.

The ASN.1 files were processed into C source code by means of the *asn1c* compiler. This step was of great consequence, as the *asn1c* compiler generates the corresponding source and header files, which define the data structures and encoding/decoding functions required for message handling in the *ms-van3t* framework. Subsequently, the files were incorporated into *ms-van3t*, thereby enabling the framework to process message structures and encoding functions in accordance with the specifications set forth by ETSI.

The integration of these files constituted the basis for the construction and extension of the *ms-van3t* framework with security features. The generated C code enabled the construction and encoding of complex message types, such as Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs). These message types are essential in the context of vehicular communication, providing real-time information about vehicle presence and safety-related incidents. Furthermore, the integration facilitated the incorporation of a security header into the existing message structure, ensuring that messages could be encoded, decoded, and verified according to the standards.

### 4.2.2 CAM profile structure and message verification

CAM (Cooperative Awareness Message) profile, previously described in Section 2.4.1, was employed to implement the security header. This process ensured that the message structure was aligned with ETSI standards and that the security header provided the necessary cryptographic protections.

- **Profile Definition:** The CAM message profile comprises a number of fields that convey information regarding the status of the vehicle, including its position, speed and heading. This data is of critical importance for ensuring the safety of vehicles and facilitating effective coordination.
- In order to enhance the functionality of the CAM message, a security header was incorporated into the existing message structure. The security header contains essential elements, such as digital signatures and certificates, which are vital for ensuring the authenticity and integrity of the message.

In this step, considering the SPDUs structure illustrated in Section 2.2.3, the following fields have been populated in accordance with the ETSI standard [15] to achieve the security profile for CAMs:

- **Signer of SignedData:** The signer field was set based on the default choice for the digest method, that correspond to the last eight bytes of the hash of the relevant certificate. The certificate was included once per second after the last inclusion of the certificate. If a CAM was received from an unknown Authorization Ticket (AT), the certificate was included in the next CAM and the timer reset accordingly.
- **Tbsdata.HeaderInfo.psid:** This field was populated with the ITS-AID value for CAMs, as specified in [19]. This ensures the CAM message is appropriately identified within the V2X communication framework.
- **Tbsdata.HeaderInfo.inlineP2pcdRequest:** This component was included under certain conditions, specifically when the ITS-S received a CAM containing a digest pointing to an unknown authorization ticket, or when the signer of the CAM was set to a certificate referencing an unknown Authorization Authority. In these cases, the certificate digests were populated in this field to inform the ITS-S of the unknown certificates.
- **RequestedCertificate:** This component was added when the ITS-S received a CAM with an inlineP2pcdRequest containing the digest of a valid Certification Authority (CA). In this situation, the requested certificate was included in the next CAM, unless a valid certificate was received before the next CAM generation. In the case of valid certificates already being present or pending, the component remained unpopulated.
- **All other components of tbsdata.HeaderInfo:** As per the standard, any additional components allowed but not specifically required were left absent, as dictated by the constraints of the CAM security profile.

Subsequently, the CAM message, inclusive of the security header, was encoded using the C structures derived from the ASN.1 specifications.

- To ascertain the efficacy of the transmission and reception processes, encoding and decoding tests were conducted. The encoded CAM message, which included the appended security header, was transmitted and subsequently decoded at the receiving end to ensure that the message was transmitted accurately. It was imperative to ascertain that the fields were duly populated, that the security header was included, and that the integrity of the data was preserved throughout the transmission process.
- A packet analysis was conducted using Wireshark, a network protocol analyser. Wireshark was employed to capture and inspect the transmitted messages at a detailed level. By analysing the packet contents, shown in Figure 4.2, it was confirmed that the CAM messages were encoded and sent correctly, and that the security header had been appended without any corruption or loss of data. This analysis provided valuable insights into the structure and transmission of V2X messages, ensuring compliance with the ETSI standards.

```

- Secured Packet
- Ieee1609Dot2Data
  protocolVersion: 3
  - content: signedData (1)
    - signedData
      hashId: sha256 (0)
      tbsData
      - signer: certificate (1)
        - certificate: 1 item
          - Item 0
            - Certificate
              version: 3
              type: explicit (0)
              issuer: sha256AndDigest (0)
              toBeSigned
                id: none (3)
                cracaId: 303030
                crlSeries: 0
                validityPeriod
                appPermissions: 2 items
              - verifyKeyIndicator: verificationKey (0)
                - verificationKey: ecdsaNistP256 (0)
                  - ecdsaNistP256: compressed-y-1 (3)
                    compressed-y-1: 6333353435383637306138313937323061646631626534336337383263326330
              - signature: ecdsaNistP256Signature (0)
                - ecdsaNistP256Signature
                  - rSig: x-only (0)
                    x-only: 6333353435383637306138313937323061646631626534336337383263326330
                    sSig: 7a33353435383637306138313937323061646631626534336337383263326330
              - signature: ecdsaNistP256Signature (0)
                - ecdsaNistP256Signature
                  - rSig: compressed-y-1 (3)
                    compressed-y-1: 6333353435383637306138313937323061646631626534336337383263326330
                    sSig: 7a33353435383637306138313937323061646631626534336337383263326330

```

**Figure 4.2:** Packet capture via wireshark

### 4.2.3 Digital Signature: Creation and Verification

The incorporation of the digital signature mechanism within the security header constituted a pivotal element in guaranteeing the integrity and authenticity of the transmitted data in V2X communications. Digital signatures serve as a primary defence against attacks such as the Sybil attack, which involves the generation

of multiple fictitious identities by malicious actors. The signature mechanism was designed in accordance with ETSI security specifications, thereby ensuring resilience against tampering and impersonation.

### Signature Creation

The process commences with the generation of a digital signature for each Cooperative Awareness Message (CAM). The diagram in Figure 4.3 delineating the methodology for signature creation outlines the following steps:

#### 1. Message Input and Hashing:

- The process starts by inputting the tbsData and certificate in hexadecimal strings. If the message is identified as a CAM, these components are converted to bytes.
- A SHA-256 cryptographic hash function is then applied to both the tbsData and certificate to produce individual hashes. These two hashes are concatenated, and the SHA-256 hash is computed once more to generate the final message digest.

#### 2. Signature Computation:

- The elliptic curve keypair (EC\_KEY) associated with the certificate is loaded to sign the message digest.
- Using the private key, a signature is computed, and the values for the r and s components of the elliptic curve signature are extracted.
- The output values of r and s are then converted to hexadecimal strings and included in the digital signature, along with the sender's certificate.

### Signature Verification

The process of digital signature verification is undertaken at the receiving end with the objective of guaranteeing both message integrity and authenticity. The diagrammatic representation of the verification process is presented in Figure 4.4 and summarized below:

1. **Input and Hash Verification:** Upon receipt of a CAM, the tbsData and certificate are once more entered in hexadecimal strings and converted to bytes. The SHA-256 hash function is applied in a manner analogous to that employed during signature creation, with the individual and concatenated hashes computed.

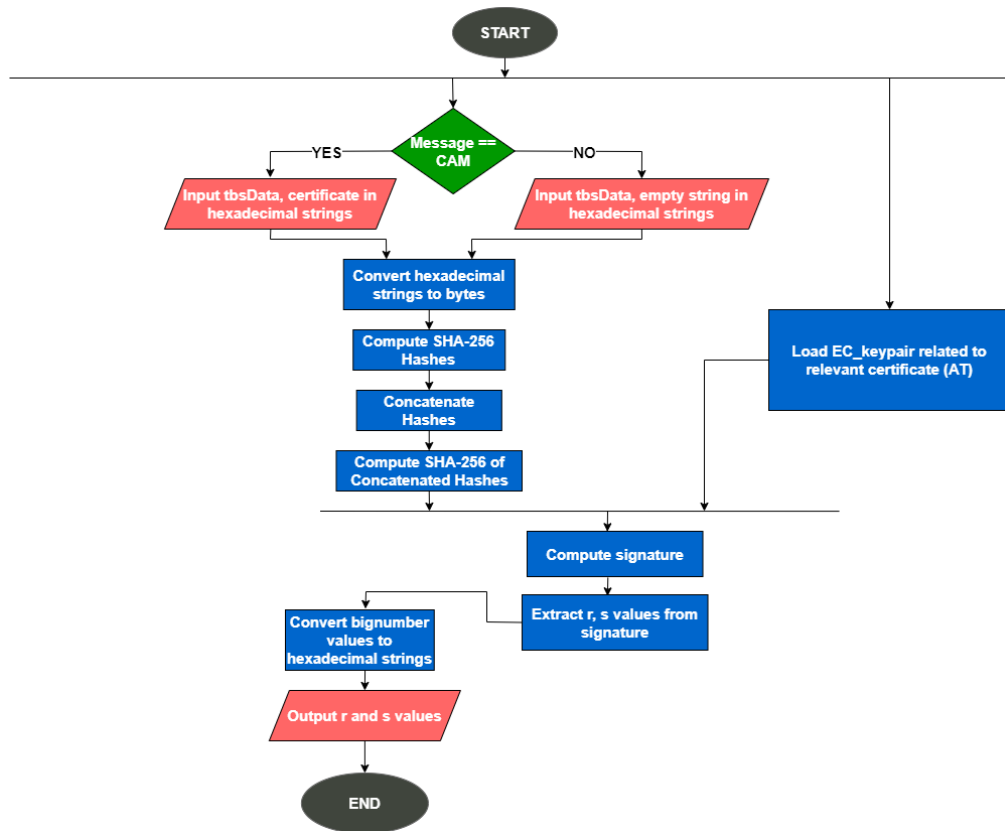


Figure 4.3: Pseudocode signature creation

## 2. Signature Verification:

- The r and s values from the signature are inputted and converted into big-number (BIGNUM) objects.
- An ECDSA signature object is created, and the public key (retrieved from the certificate) is used to perform the verification.
- The public key is first compressed, and its size is checked to ensure it is 33 bytes. The elliptic curve key (EC\_KEY) is reconstructed using the point representation of the public key.

## 3. Validation:

- If the signature verification process determines that the hash generated from the received message matches the signature, the signature is deemed valid, and the message is confirmed to have originated from a legitimate source without being altered.

- If the signature does not match, an error is printed, indicating the signature's invalidity.

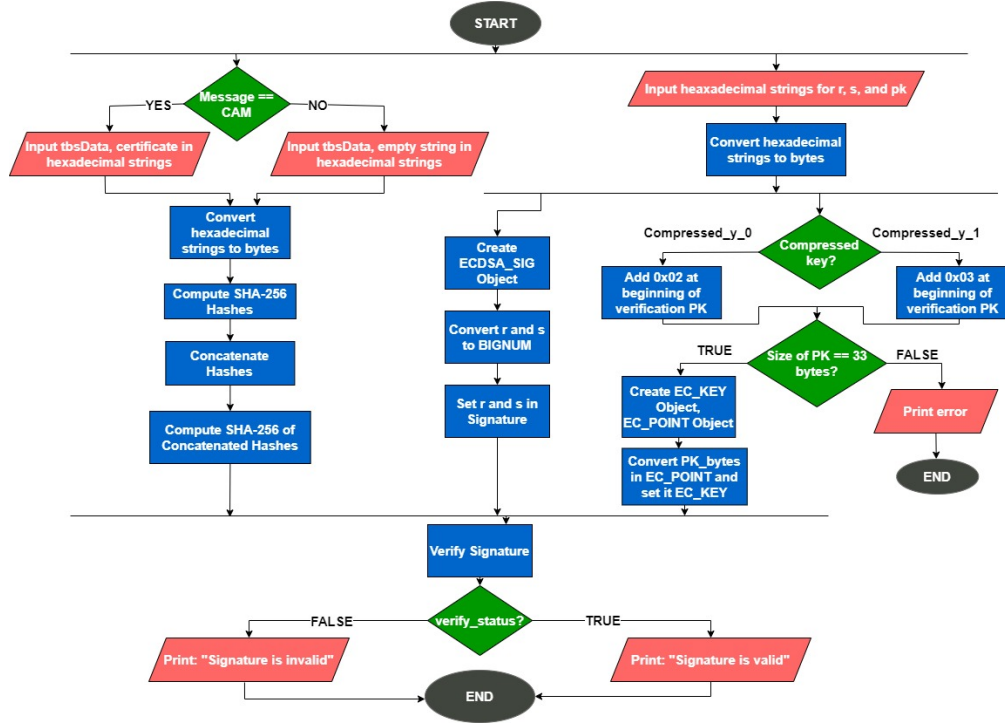


Figure 4.4: Pseudocode signature verification

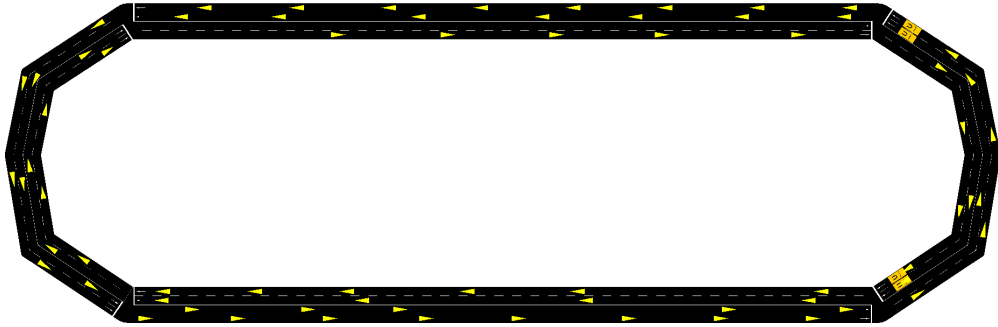
This dual-layered process guarantees the authentication of the sender and the integrity of the transmitted message, effectively preventing unauthorised message tampering and Sybil attack attempts. The process of generating and verifying the digital signature was subjected to rigorous testing within the ms-van3t framework, ensuring compliance with ETSI standards and confirming that these security measures do not introduce latency that could interfere with real-time V2X communications.

### 4.3 Simulation results

This section presents an analysis of the simulation results for two critical performance metrics: average latency and packet reception ratio (PRR). These metrics are crucial for assessing the effectiveness and reliability of the V2X communication system, especially under the influence of security mechanisms. The simulations were performed over a range of vehicle densities, from 10 to 100 vehicles, with increments

of 10 vehicles per scenario. Each scenario was simulated twice: once with security mechanisms enabled and once without. This dual setup allows for a comprehensive comparison to evaluate the impact of security on system performance as vehicle density increases.

For example, Figure 4.5 shows the simulation performed in ms-van3t with a scenario of 70 vehicles.

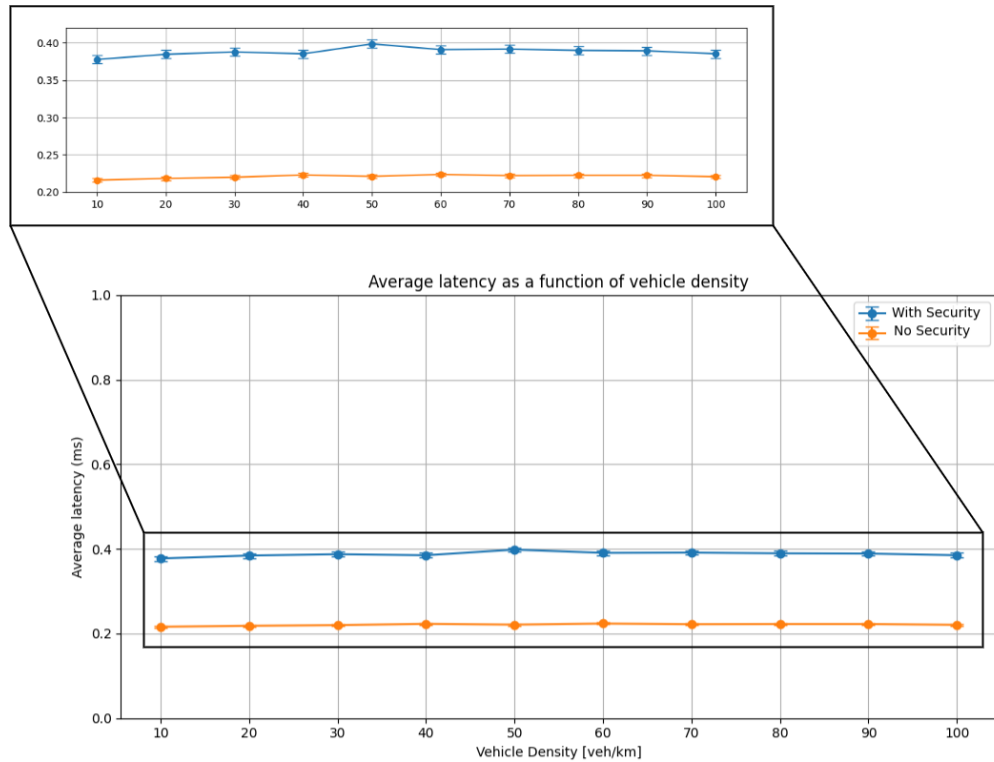


**Figure 4.5:** ms-van3t simulation scenario

### 4.3.1 Average Latency

Figure 4.6 illustrates the mean latency as vehicle density varies from 10 to 100 vehicles per kilometre. This metric is pivotal for assessing the efficacy of message transmission across vehicles in real-time V2X communication, particularly for safety-critical applications such as collision avoidance, intersection management, and autonomous vehicle coordination, where low latency is of paramount importance.

- **No Security (orange curve):** It is noteworthy that the system devoid of security measures consistently exhibits low latency, with values hovering around 0.2 ms across all vehicle densities. This outcome aligns with the hypothesis that the absence of security-related overheads minimises processing time, thereby facilitating faster message transmission. Nevertheless, while the unsecured system offers accelerated communication, it is devoid of the safeguards essential for secure V2X communications, rendering it susceptible to attacks and other threats that could potentially compromise the reliability and safety of the system.
- **With Security (blue curve):** The latency is observed to start at approximately 0.35 ms and exhibits slight fluctuations as vehicle density increases, reaching values approaching 0.4 ms. This demonstrates that although security



**Figure 4.6:** Average latency as a function of vehicle density

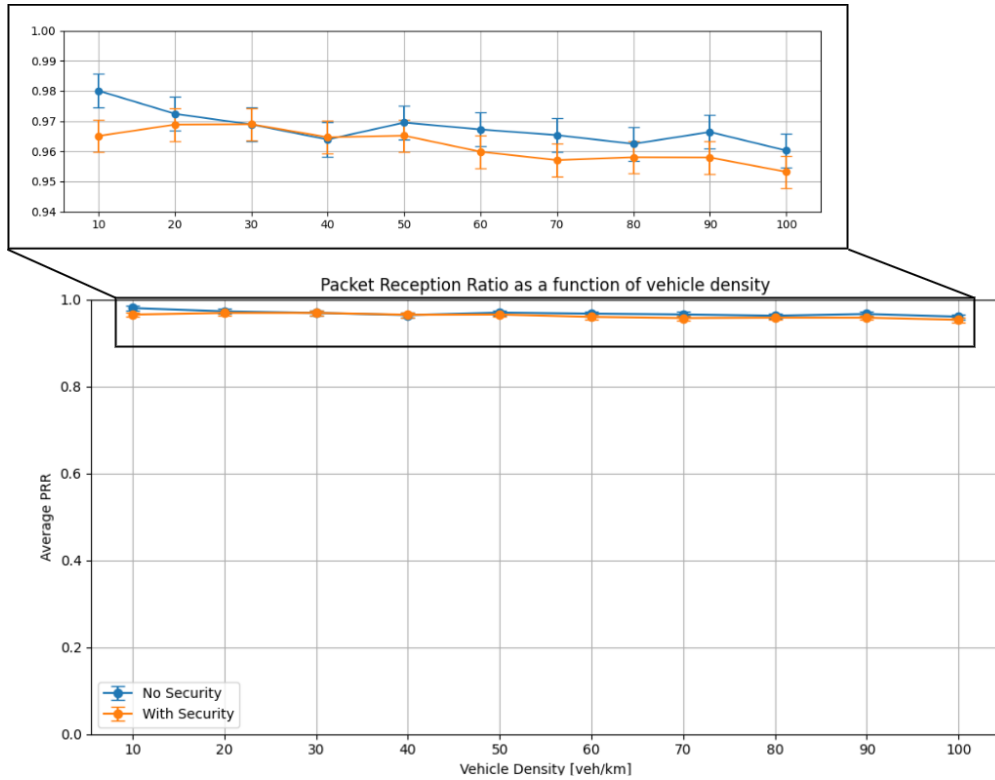
mechanisms, such as encryption, decryption and digital signature verification, introduce a certain degree of overhead, their impact on latency is relatively minor. The system’s capacity to maintain low latency, even with security enabled, indicates the effectiveness of cryptographic implementations and the optimisation of secure message handling. This is a favourable outcome, as it demonstrates that the system can uphold rigorous security requirements without significantly compromising performance, ensuring timely communication in V2X scenarios.

The results of the latency tests are of great significance for the field of V2X communication, where low latency is of paramount importance, particularly in the context of safety-critical functions. V2X standards, such as IEEE 802.11p and C-V2X, define strict latency requirements with the objective of ensuring the rapid delivery of emergency alerts and hazard notifications. The results demonstrate that the secured system is capable of maintaining acceptable latency levels even when subjected to increased vehicle density, indicating its suitability for real-world deployment in scenarios where both security and performance are paramount. The consistently low latency observed in the unsecured system highlights the inherent



trade-off between security and performance. However, in safety-critical V2X environments, the benefits of secure communication outweigh the slight increase in latency observed in the secured system.

### 4.3.2 Packet Reception Ratio (PRR)



**Figure 4.7:** Packet Reception Ratio (PRR) as a function of vehicle density

Figure 4.7 illustrates the Packet Reception Ratio (PRR) as vehicle density increases. The PRR is a pivotal metric in assessing the dependability of a communication system in vehicular networks. It gauges the proportion of successfully received packets out of the total transmitted packets. Elevated PRR values signify dependable communication, which is vital for V2X systems, where uninterrupted and precise message delivery is essential for both safety and operational efficacy.

- **Without Security (blue curve):** The PRR starts at a relatively high level (approximately 0.98) and remains relatively stable with only minor fluctuations as vehicle density increases. These fluctuations are likely to result from increasing channel contention or packet collisions at higher densities. However, the PRR never falls below 0.95, which indicates that the system is

capable of maintaining a high degree of communication reliability, even as the number of vehicles and, consequently, the communication load increases.

- **With Security (orange curve):** As with the unsecured scenario, the PRR remains high, though at a slightly lower level than the PRR without security, remaining above 0.95 even at the highest vehicle densities. This slight decrease is to be expected, as the introduction of security measures, such as message encryption and authentication, can result in minor overheads that may increase the probability of packet loss due to increased processing time or missed transmission windows. However, the difference is minimal, suggesting that the impact of security mechanisms on communication reliability is negligible in this case.

These elevated PRR values are encouraging with respect to V2X communication, particularly in scenarios where vehicles are required to continuously exchange safety-critical information. A high PRR ensures that the majority of transmitted messages are successfully received, which is crucial for ensuring the consistent delivery of important notifications such as accident warnings, traffic updates, and road hazard alerts. The high reliability observed in both the secured and unsecured systems indicates that the communication protocol is sufficiently robust to handle the challenging environment of vehicular networks, where rapid changes in topology and high-speed mobility can affect communication quality.

### 4.3.3 Analysis

The aforementioned results illustrate that the communication system exhibits optimal performance with regard to both latency and reliability. The average latency remains low in both secured and unsecured scenarios, even as the density of vehicles on the road increases. This is a pivotal outcome for V2X communication, where low latency is essential to guarantee the prompt delivery of time-sensitive messages, thereby enhancing vehicle and road safety. It is noteworthy that the security mechanisms do not result in significant delays, indicating that the system can provide security without compromising performance. This is an important feature for future V2X systems, particularly in light of the increasingly stringent regulatory and safety requirements.

Similarly, the high PRR observed across all vehicle densities indicates that the communication system is highly reliable, even in scenarios with a large number of vehicles. This is vital for V2X systems, where packet loss could result in missed safety-critical information. The fact that the PRR remains above 95% even with security enabled suggests that the system can handle both the traffic load and security overhead effectively, ensuring a consistent and reliable communication flow.

In conclusion, the results demonstrate that the V2X communication system is capable of providing both low latency and high reliability, even with security mechanisms in place. This makes it an appropriate choice for deployment in real-world V2X scenarios, where both performance and security are essential for ensuring safe and efficient vehicle communication.

# Chapter 5

## OScar

The OScar framework (Open Stack for Car), as outlined in reference [20], is an open-source, ETSI-compliant C-ITS (Cooperative Intelligent Transport Systems) stack designed for deployment in real-world vehicular field testing on embedded hardware devices. As a lightweight, self-contained solution, OScar is well-suited for practical, non-simulated environments, as it supports V2X (vehicle-to-everything) communication protocols. It is imperative that the software stack in question be customisable when testing V2X systems in such settings. OScar facilitates this by providing comprehensive ETSI-compliant services on embedded platforms.

The key ETSI services supported by OScar include Cooperative Awareness Messages (CAMs), Vulnerable Road User Awareness Messages (VAMs), and Decentralized Environmental Notification Messages (DENMs). These message types facilitate communication between vehicles and infrastructure, thereby enhancing road safety and traffic management. Furthermore, OScar is optimised for low-power, cost-effective embedded devices, such as those running Linux-based distributions such as OpenWrt. This enables a bridge between theoretical simulations and practical deployments, and provides researchers with tools to test V2X systems in realistic driving conditions.

### 5.1 Code Porting to the OScar Framework

The preliminary development of V2X security features was conducted within the ms-van3t simulation environment, which offered invaluable insights but was constrained to simulated scenarios. In order to assess the resilience of these security protocols in real-world operational contexts, the code was adapted to the OScar framework. The porting process was relatively straightforward, as both ms-van3t and OScar have a similar codebase structure, with numerous parallels between the models used in ms-van3t and those implemented in the Linux kernel.

The principal objective of the porting process was to facilitate a transition from a simulation-based environment that handled scenarios with up to 100 vehicles to a framework that manages a single, real-world on-board unit (OBU) for live vehicular testing. The transition to OScar permits the field testing of V2X communication protocols in realistic conditions involving actual vehicles and environmental factors. These elements, which cannot be fully replicated in simulations, include traffic density, physical obstacles, and weather conditions.

The modifications to the code were designed to enhance its suitability for embedded hardware. The compatibility of OScar with a range of hardware, including that supporting the IEEE 802.11p vehicular communication protocol, enabled the conduct of live field tests in which vehicles could interact with each other and roadside infrastructure via CAMs. This capability provides a robust basis for the assessment of V2X protocols in real-world settings.

## 5.2 Test Methodology

Following the adaptation of the code to integrate with the OScar framework, a comprehensive testing methodology was developed to assess the latency, conformity, and robustness of V2X message handling under both real-world and controlled conditions. The testing was conducted in two phases. Initially, the system's performance was evaluated in an active vehicular environment, which was followed by server-based testing. This allowed for repeatable, controlled simulations to be conducted using stored GPS trace data.

In the field tests, the OBU was installed in a moving vehicle to observe system behaviour under variable environmental conditions, such as urban landscapes with potential signal obstructions. In contrast, server testing used identical software configurations to replay field test data in a stable environment, where external interference factors were minimised. This two-phase approach enabled a direct comparison between performance metrics collected in the field and those obtained in a controlled setting, providing insights into how well the system could handle live operational demands versus simulated ones.

### 5.2.1 Field Testing Setup

In order to evaluate the efficacy of the V2X communication model in actual operational conditions, preliminary field trials were conducted utilising an On-Board Unit (OBU) installed in a vehicle. The OBU, equipped with an AMD Embedded G-Series GX-412TC CPU, was configured to transmit and receive Cooperative Awareness Messages (CAMs) while in movement. This phase of testing enabled the measurement of latency, signal integrity, and resilience under conditions analogous to those anticipated in actual deployment scenarios.

The configuration of the tests involved the integration of the OBU hardware into the vehicle, alongside the GPS, antenna, and communication modules, as illustrated in Figures 5.1, 5.2, 5.3 and 5.4. This configuration replicated the circumstances of an urban driving context, wherein V2X-equipped vehicles are required to transmit and receive messages with minimal latency and high reliability. The vehicle traversed routes encompassing diverse levels of environmental interference, including high-density urban areas and open spaces, enabling the observation of potential influences on packet transmission quality and timing.



**Figure 5.1:** OBU installed in vehicle



**Figure 5.2:** TX and RX of CAMs



**Figure 5.3:** Power Station



**Figure 5.4:** Antenna

For each transmitted CAM, it was verified message conformity to ETSI standards using Wireshark, illustrated in Figure 5.5.

The packets captured from Oscar and illustrated in Figures 5.7 and 5.6, respectively, were compared with those generated by `ms-van3t` in a simulated environment. Figure 5.7 depicts packets captured for CAM with a *signerIdentifier* equal to the digest, while Figure 5.6 depicts packets captured for CAM with this field equal to

No.	Time	Source	Destination	Protocol	Length	Info
5829	16.955122296	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
5986	17.355975403	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
6091	17.755230399	0.5.0.00:25:90:bc:4..	Broadcast	CAM	341	CAM
6215	18.155077906	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
6285	18.555078362	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
6428	18.955257219	0.5.0.00:25:90:bc:4..	Broadcast	CAM	341	CAM
6552	19.355095815	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
6649	19.755176662	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
6875	20.155243599	0.5.0.00:25:90:bc:4..	Broadcast	CAM	341	CAM
7061	20.555121885	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
7185	20.955183762	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
7284	21.255300625	0.5.0.00:25:90:bc:4..	Broadcast	CAM	341	CAM
7343	21.555182798	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
7404	21.855127710	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM
7448	22.155112103	0.5.0.00:25:90:bc:4..	Broadcast	CAM	192	CAM

```

Frame 145: 192 bytes on wire (1536 bits), 192 bytes captured (1536 bits) on interface eno1, id 0
Ethernet II, Src: SuperMic_bc:43:32 (00:25:90:bc:43:32), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- GeoNetworking
  - Basic Header
  - Secured Packet
    - Ieee1609Dot2Data
      - protocolVersion: 3
      - content: signedData (1)
        - signedData
          - Common Header
          - Topologically-Scoped Broadcast Packet
            - ...
0000 ff ff ff ff ff ff 00 25 90 bc 43 32 89 47 12 00  ....%..C2.G..
0010 50 01 03 81 00 40 03 80 51 20 50 02 80 00 2d 01  P...@..QP.....
0020 00 14 00 00 25 90 bc 43 32 2c ae 8d 7e 1a d4 a9  ...%.C2,.....
    
```

Figure 5.5: Wireshark capture of CAMs

the certificate. The comparison revealed that the packet structures were consistent across both environments, thereby confirming that the ported code retained its integrity. Wireshark also enabled the real-time analysis of the secure packet transmission, confirming that Oscar was effectively generating the secured packets as anticipated. This process is crucial for validating the system’s functionality prior to conducting further tests in a field environment.

```

- Secured Packet
  - Ieee1609Dot2Data
    - protocolVersion: 3
    - content: signedData (1)
      - signedData
        - hashId: sha256 (0)
        - tbsData
          - signer: certificate (1)
            - certificate: 1 item
              - Item 0
                - Certificate
                  - version: 3
                  - type: explicit (0)
                  - issuer: sha256AndDigest (0)
                  - toBeSigned
                    - id: none (3)
                    - cracaId: 303030
                    - crlSeries: 0
                    - validityPeriod
                    - appPermissions: 2 items
                      - Item 0
                        - PsidSsp
                          - psid: psid-ca-basic-services (36)
                          - ssp: bitmapSsp (1)
                      - Item 1
                        - verifyKeyIndicator: verificationKey (0)
                        - verificationKey: ecdsaNistP256 (0)
                    - signature: ecdsaNistP256Signature (0)
                    - ecdsaNistP256Signature
                      - rSig: x-only (0)
                        - x-only: 6333353435383637306138313937323061646631626534336337383263326330
                        - sSig: 7a33353435383637306138313937323061646631626534336337383263326330
                    - signature: ecdsaNistP256Signature (0)
            
```

Figure 5.6: CAM with SignerIdentifier equal to certificate

```

- Secured Packet
- IEEE1609Dot2Data
  protocolVersion: 3
  content: signedData (1)
  - signedData
    hashId: sha256 (0)
    tbsData
    - payload
      data
        protocolVersion: 3
        content: unsecuredData (0)
        unsecuredData: 20500200002d01001400dc2c6e3d4119ef6970a91adc53ef049116640000007800000000...
    headerInfo
  - signer: digest (0)
    digest: 3165376437653831
  - signature: ecdsaNistP256Signature (0)
    - ecdsaNistP256Signature
      rSig: x-only (0)
      sSig: ee4e672a9b5b360dea9c7e9b5d49cf87cd1d87b50bd92eae9b89201926338a2b

```

**Figure 5.7:** CAM with SignerIdentifier equal to digest

The field-based setup provided a baseline for evaluating the performance of the OBU hardware and software in handling CAMs under conditions where factors such as processing power, physical obstructions, and real-time signal dynamics could potentially impact latency and packet integrity. These real-world results subsequently informed the controlled server test comparisons, where variables such as signal interference were minimised.

## 5.2.2 Server Testing Setup with GPS Trace Replay

To conduct a comparative analysis, the identical GPS trace utilized in field tests was replayed on a server. This configuration included an AMD EPYC 7601 CPU, which provided enhanced processing capabilities to evaluate discrepancies in latency and processing stability. This setting facilitated a systematic assessment of packet handling, circumventing the potential confounding effects of physical variables (e.g. signal interference and movement).

## 5.3 Results

The results presented here detail the metrics studied across the two setups and the analysis of their impact on V2X communication performance. The main metrics under observation included mean latency and variance for each packet type across transmission (TX) and reception (RX) scenarios.

### 5.3.1 Metrics and Formulas Used

The key statistical metrics, namely mean latency and variance, were calculated using the following formulas. Mean latency  $\mu$  for each message type is:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i$$



where  $x_i$  represents the latency of each message and  $N$  is the total number of latency measurements for the message type.

The variance  $\sigma^2$  for each message type, which captures the dispersion of latency values, is calculated as:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$$

These metrics provide insight into the processing stability and efficiency of each CPU under test conditions, which are essential for ensuring robust V2X performance.

### 5.3.2 Latency Data from Server and Field Tests

The following tables present the latency values obtained from the server and field tests.

Operation	Message Type	Mean Latency ( $\mu\text{s}$ )	Variance ( $\mu\text{s}^2$ )
TX	Certificate	273.16	1724.89
TX	Digest	101.14	608.47
RX	Certificate	1906.17	108751.84
RX	Digest	1758.29	19799.63

**Table 5.1:** Latency Results for Server Test

Operation	Message Type	Mean Latency ( $\mu\text{s}$ )	Variance ( $\mu\text{s}^2$ )
TX	Certificate	2042.58	29712.36
TX	Digest	835.85	10587.78
RX	Certificate	7273.26	603342.34
RX	Digest	6751.25	1730466.29

**Table 5.2:** Latency Results for Field Test

### 5.3.3 Analysis of Latency and Performance Impact of CPU Differences

Upon examining the mean latency and variance values across the two environments, several notable trends emerge:

- TX Latency on Server vs Field Test: The mean latency for certificate-based messages on the server is 273.16  $\mu\text{s}$ , which is a markedly lower value than that

observed in the field test, which was 2042.58  $\mu\text{s}$ . With regard to digest-based messages, the server latency is 101.14  $\mu\text{s}$ , while the field test latency is 835.85  $\mu\text{s}$ . This suggests that the superior CPU of the server, namely the AMD EPYC 7601, enables faster packet processing due to its higher processing power.

- **RX Latency on Server vs Field Test:** The mean RX latency for certificate-based messages on the server is 1906.17  $\mu\text{s}$ , which is significantly lower than the 7273.26  $\mu\text{s}$  observed in field testing. Similarly, digest-based messages also demonstrate a discrepancy in latency, with the server test exhibiting 1758.29  $\mu\text{s}$  in comparison to the field test's 6751.25  $\mu\text{s}$ .
- **Variance Comparison:** Once more, the variance in latency is lower in the server test than in the field test, particularly for certificate-based messages. To illustrate, the TX variance for certificate messages on the server is 1724.89  $\mu\text{s}^2$ , whereas it is 29712.36  $\mu\text{s}^2$  in the field test. These variances indicate that the server's AMD EPYC 7601 CPU offers more consistent processing times.
- **CPU Performance Impact:** The diminished latency and variance values observed in server tests underscore the significance of CPU performance in V2X systems. The enhanced core count and clock speed of the AMD EPYC 7601 result in superior stability and efficiency in packet processing, which is indispensable for dependable V2X communications.

The results highlight the importance of optimising hardware, particularly in the context of field deployments of V2X systems, where stability and low latency are crucial for the safe and efficient handling of messages. Furthermore, the comprehensive testing has demonstrated that the implemented security features, particularly the authentication mechanisms, function effectively under real-world conditions. This validation not only affirms the integrity of the V2X communication protocols but also ensures that the system can reliably secure the exchange of Cooperative Awareness Messages (CAMs). Overall, the findings indicate that the transition from a simulated environment to practical deployment has been successfully achieved, laying a solid foundation for future advancements in cooperative intelligent transport systems.

## 5.4 PKI Integration and Real Certificate Management

In order to guarantee compliance with the ETSI standards and enable the secure transmission of V2X messages, a transition was initiated from the utilisation of self-created fictitious certificates, which were employed during the initial simulations, to

the generation of authentic certificates through a live PKI infrastructure. This step was of great significance in enabling the transition from theoretical development to a practical, secure deployment. In the real-world deployment phase, collaboration with EVIDEN facilitated access to their PKI portal, thereby enabling the registration of the ITS station utilized in the testing environment and the acquisition of valid authorization and enrollment credentials. The process of PKI integration was carried out as follows (for further technical details of the ETSI-defined message exchange sequence, refer to Section 2.3.1):

1. **ITS Station Registration and Profile Setup:** The ITS station that was used for testing was duly registered on the PKI portal. During this phase, the profile of the station was configured, incorporating key identifiers and operational particulars. This registration guaranteed that the station could interact with the Enrollment Authority (EA) and Authorization Authority (AA) in accordance with the ETSI security model. Furthermore, access points, such as URLs and the associated public certificates for both EA and AA, were also retrieved and integrated into the system configuration.
2. **EC Request procedure:** The procedure for the generation and transmission of Enrollment Certificate (EC) requests was implemented using standalone C++ code and additional supporting scripts, as for the encryption algorithm that can be seen in Appendix A, reference B.5. The EC request included all the mandatory fields as defined in the ETSI standards and illustrated in previous chapters in Figure 2.13, including identifiers for the ITS station and cryptographic material generated during runtime. The cryptographic material comprised public-private key pairs, generated through the utilisation of secure elliptic curve algorithms. Subsequently, the constructed EC request was transmitted to the Enrollment Authority (EA) via an HTTPS POST method with a binary-encoded payload. The secure channel ensured the protection of sensitive cryptographic information during transmission. Upon successful validation of the request, the EA returned an EC response, illustrated in Figure 2.14, containing the issued Enrollment Certificate. This response was decoded locally using the C++ code, and the extracted EC was stored securely for use in subsequent operations. This process established the first layer of the PKI hierarchy, enabling the ITS station to authenticate itself to the Authorization Authority.
3. **Creation and Submission of AT Requests:** On the basis of the Enrollment Certificate (EC), obtained from the EA, the generation of an Authorization Ticket (AT) request procedure was devised, the details of which are illustrated in Figure 2.15. At this point, a new set of cryptographic material was generated and included in the request. The AT request linked the new credentials with

the existing EC, thereby ensuring the integrity of the chain of trust between the ITS station, the EA, and the Authorisation Authority (AA). The AT request was transmitted to the AA via an HTTPS POST method, accompanied by a binary payload. The AA proceeded to validate the request by decrypting the payload and performing all necessary security checks. These included verifying the digital signature and contacting the EA to confirm the validity of the EC. Upon successful validation, the AA issued the authorisation ticket (AT) in binary format. The response, which exhibits the structure illustrated in Figure 1, was decoded with the aid of the standalone C++ code, and the extracted AT was stored in a secure location. The AT is a critical credential, as it enables the ITS station to digitally sign V2X messages, such as Cooperative Awareness Messages (CAMs), with a genuine digital certificate. To guarantee compatibility with the OScar framework, the decoded AT was structured to align with the certificate format anticipated by the framework. This integration permitted the ITS station to transmit authentic CAMs, which were authenticated by a verifiable PKI infrastructure

The incorporation of authentic certificates signified a significant advancement from the simulated environment, facilitating the establishment of a fully operational and secure V2X communication system. By utilising authentic certificates and adhering to the established PKI workflow, the system achieved an elevated level of compliance and preparedness for rigorous real-world field testing.

# Chapter 6

## Conclusion

The development of Vehicle-to-Everything communications represents a fundamental shift in the way vehicles interact with their environment. This thesis focused on addressing the security challenges of V2X systems by implementing cryptographic solutions, evaluating performance implications, and exploring the interoperability of secure messaging protocols across V2X technologies. Through analysis and implementation, it highlighted the critical role of security, particularly in protecting vehicle and infrastructure communications from various cyber threats such as Sybil attacks, message tampering and denial of service attacks.

### 6.1 Key contributions

This thesis provides several contributions to the field of V2X communication:

- **Implementation of cryptographic mechanisms:** By integrating Public Key Infrastructure (PKI) and Elliptic Curve Digital Signature Algorithm (ECDSA), the thesis provides a security framework that strengthens the integrity and authenticity of V2X messages. These cryptographic mechanisms enable reliable vehicle authentication and message verification in an open, decentralised vehicular network.
- **Evaluation of IEEE and ETSI standards:** The work leverages the IEEE 1609.2 and ETSI standards for securing V2X communications, detailing their applicability, limitations and interoperability. By examining protocols such as CAM, the thesis illustrates the need for robust security standards that are also adaptable to the demands of high-speed, low-latency V2X environments.
- **Performance evaluation through simulation and field testing:** Through simulations in the ms-van3t and OScar frameworks, the thesis evaluates the impact

of security protocols on latency and packet reception ratios. The On-Board Unit (OBU) field tests assess the real-world latency impact of secure CAMs and provide data on the feasibility of deploying these protocols in different traffic and infrastructure setups.

- Formal verification of security properties: Using formal tools such as ProVerif, the work validated the implemented security solutions against potential vulnerabilities. The verification provides confidence in the accessibility, authentication and confidentiality properties essential for V2X systems.
- Obtaining EC and AT: The acquisition of an Enrollment Certificate (EC) from the Enrollment Authority (EA), which is then used to request an Authorization Ticket (AT) from the Authentication Authority (AA), marks the conclusion of the thesis. This process enables the secure signing of Cooperative Awareness Messages (CAMs) within the V2X network, ensuring proper authentication and authorization.

## 6.2 Future works

This thesis has successfully addressed essential security aspects of V2X communication, including authentication and data integrity. However, addressing privacy concerns is equally critical for the widespread adoption and viability of vehicular networks. Privacy in V2X can be divided into two main dimensions: identity privacy and location privacy. Identity privacy ensures that the identity of the sender remains hidden, while location privacy prevents the tracking of a vehicle's geographical position [21].

To enhance identity privacy, future research could explore methods that provide conditional anonymity and unlinkability. Conditional anonymity would allow the true identity of the sender to remain hidden from other vehicles and entities, except in certain controlled circumstances where a Trusted Authority (TA) could reveal the identity if necessary. Achieving unlinkability would prevent observers from tracing multiple messages back to the same sender. This could be achieved through group-based signature schemes, where each group member uses a shared group key to sign messages anonymously, making it difficult to link signatures to individual identities.

Alternatively, identity-based schemes using pseudonyms offer another effective approach to identity protection. Vehicles could frequently change pseudonyms generated with random values to avoid traceability. Future research could consider mechanisms to manage the high pseudonym consumption rate in V2X environments, such as self-generated pseudonyms or those provided periodically by Roadside Units (RSUs) or the TA. Vehicles could even pre-store a significant number of pseudonyms

to ensure sufficient anonymity over longer journeys, which is feasible given modern storage capabilities.

Location privacy could be enhanced by integrating encryption schemes that prevent unauthorised entities from identifying the location of the sender. Advanced encryption schemes could allow vehicles to exchange necessary information while maintaining spatial anonymity.

Incorporating these privacy-preserving techniques would broaden the security framework of V2X communications, ensuring that vehicles can communicate securely and privately, meeting the full range of modern vehicular network security requirements. Emphasising privacy not only strengthens the security architecture, but also promotes public trust and aligns with regulatory standards, supporting the sustainable growth of V2X technologies in intelligent transport systems.

# Appendix A

## Proverif C-ITS trust model

This appendix presents relevant portions of the ProVerif code used to model the C-ITS trust system, with a particular focus on key processes such as enrollment, message verification, and authentication within the ITS network. The code snippets demonstrate a range of cryptographic operations, including signature validation and event handling for various stages of communication.

### A.1 Standard Scheme of C-ITS Trust Model

#### A.1.1 Public and Private Key Definitions

Defines the cryptographic types used for keys and results of signature checks.

```
type pkey.                (* public key *)
type skey.                (* private key *)
type keymat.              (* key material *)
type result.              (* result of check signature *)
```

#### A.1.2 Request Functions

Illustrates how encrypted requests are structured and how keys are extracted from them.

```
fun request(bitstring,bitstring) : bitstring.
reduc forall sKey_enc:bitstring,req_enc:bitstring;
  getsKenc(request(sKey_enc,req_enc)) = sKey_enc.
reduc forall sKey_enc:bitstring,req_enc:bitstring;
  getReq(request(sKey_enc,req_enc)) = req_enc.
```



## A.2 Public-Key Cryptography

### A.2.1 Encryption and Decryption

Defines public-key encryption and decryption operations, ensuring that encrypted data can be decrypted using the corresponding private key.

```

fun penc(bitstring, pkey): bitstring.
fun pk(keymat): pkey.
fun sk(keymat): skey.
reduc forall x:bitstring, y:keymat; pdec(penc(x,pk(y)),
  sk(y)) = x.

```

## A.3 Processes in the Trust Model

### A.3.1 AA Process

Key operations in the Authorization Authority (AA) process include receiving and decrypting an AT request, validating signatures, and forwarding the request to the Enrollment Authority (EA).

```

let AA(kpAA:keymat, pkITS:pkey, pkEA:pkey) =
  (* Receiving and decrypting AT request *)
  in(c, req:bitstring);
  let enc_nk2 = getsKenc(req) in
  let at_req = getReq(req) in
  let nonce_key2 = pdec(enc_nk2, sk(kpAA)) in
  let (n2, k2) = nonce_key2 in
  let outerREQ = sdec(at_req, k2) in
  ...
  if checksign(outerREQ, pk_vaa) = ok() then
    ...
    out(c, request(penc((n3,k3), pkEA), senc(sign(
  signature, sk(kpAA)), k3)));

```

### A.3.2 EA Process

The Enrollment Authority (EA) validates EC requests and manages certificate generation for ITS-S. It checks the signature and ensures the authenticity of the requester.

```

let EA(kpEA:keymat, pkITS:pkey, pkAA:pkey) =
  (* Verifying EC request *)
  in(c, req:bitstring);
  let enc_nk = getsKenc(req) in
  let ec_req = getReq(req) in
  let nonce_key = pdec(enc_nk, sk(kpEA)) in
  ...
  if checksign(innerREQ, pk_vea) = ok() then
    event EAend_EC(k1, pk_vea, pkITS);
    ...
    out(c, senc(sign((RES_CODE_OK, (n1, EC_certificate)),
  sk(kpEA)), k1));

```

### A.3.3 ITS-S Process (Initiating Request)

Shows the ITS-S process initiating an enrollment request and an AT request to the AA and EA.

```

let ITS_S(kpITS:keymat, pkAA:pkey, pkEA:pkey) =
  (* Enrolment request *)
  new k1:bitstring;
  new itsID:bitstring;
  event ITS_Sbegin_EC(k1, pk(kp_vea), pk(kpITS));
  out(c, request(penc((n1, k1), pkEA), senc(sign(sign(
  InnerECreq(itsID, pk(kp_vea)), sk(kp_vea)), sk(kpITS)
  ), k1)));
  ...
  (* AT request *)
  new k2:bitstring;
  event ITS_Sbegin_AT(k2, pk(kp_vaa), pk(kpITS), keyTag)
  ;
  out(c, request(penc((n2, k2), pkAA), senc(sign(
  InnerATreq(pk(kp_vaa), hmac_key, keyTag, sign(m, sk(
  kp_vea))), sk(kp_vaa)), k2)));

```

## A.4 Security Queries

### A.4.1 Reachability Queries

Specifies queries to ensure that key events such as the conclusion of AA, EA, and ITS processes are reachable within the protocol model.

```
query event(endAA()).
query event(endEA()).
query event(endITS_S()).
query event(endITS_R()).
```

### A.4.2 Secrecy and Authentication Queries

Tests for secrecy of keys and certificates and ensures that authentication events between the entities are valid.

```
query attacker(new k1).
query attacker(new EC_certificate).
query attacker(new AT_certificate).

event ITS_Sbegin_EC(bitstring, pkey, pkey).
event EAend_EC(bitstring, pkey, pkey).
query x:bitstring, y:pkey, z:pkey; event(EAend_EC(x, y, z))
  ==> event(ITS_Sbegin_EC(x, y, z)).

event ITS_Sbegin_AT(bitstring, pkey, pkey, bitstring).
event AAend_AT(bitstring, pkey, pkey, bitstring).
query x:bitstring, y:pkey, z:pkey, w:bitstring; event(
  AAend_AT(x, y, z, w)) ==> event(ITS_Sbegin_AT(x, y, z, w)).
```

### A.4.3 Event and Query Extensions for AT Provisioning

The provisioning process introduces new events specific to AT download requests and certificate management:

```
event ITS_Sb_ATDownload(bitstring, bitstring, pkey, pkey).
event EAe_ATDownload(bitstring, bitstring, pkey, pkey).
query x, y, z, w; event(EAe_ATDownload(x, y, z, w)) ==> event(
  ITS_Sb_ATDownload(x, y, z, w)).
```

# Appendix B

## Security Header

### B.1 GeoNetworking Integration

#### B.1.1 CreateSecurePacket

This subsection delineates the methodology for incorporating a secure packet between the common and basic headers.

```
dataRequest.data->AddHeader (commonHeader);  
if(enableSecurity){  
    dataRequest = m_security->createSecurePacket (  
dataRequest);  
}  
dataRequest.data->AddHeader (basicHeader);
```

In this snippet, the `dataRequest` first adds a `commonHeader`. If security is enabled, the function `createSecurePacket` is invoked, wrapping the packet with necessary security mechanisms such as encryption or digital signatures. Afterward, the `basicHeader` is appended.

#### B.1.2 CreateSecurePacket Function Definition

This function handles the creation of the secured packet. It processes the input packet, adds cryptographic operations, and returns the secured version.

```
GNDataRequest_t Security::createSecurePacket (  
    GNDataRequest_t dataRequest)  
{  
    auto ieeeData = asn1cpp::makeSeq (Ieee1609Dot2Data);  
    // IeeeData, protocol version
```

```

asn1cpp::setField (ieeeData->protocolVersion,
    m_protocolVersion);
// IeeeContent
auto ieeeContent = asn1cpp::makeSeq (
    Ieee1609Dot2Content);
asn1cpp::setField (ieeeContent->present,
    Ieee1609Dot2Content_PR_signedData);
// SignedData part
auto signData = asn1cpp::makeSeq (SignedData);

.....

auto dataContentPayload = asn1cpp::makeSeq (
    Ieee1609Dot2Content);
asn1cpp::setField (dataContentPayload->present,
    Ieee1609Dot2Content_PR_unsecuredData);
// Copy data request in buffer
uint8_t *buffer;
buffer = (uint8_t *) malloc ((dataRequest.data->
    GetSize ()) * sizeof (uint8_t));
dataRequest.data->CopyData (buffer, dataRequest.data->
    GetSize ());
std::string packetContent ((char *) buffer, (int)
    dataRequest.data->GetSize ());
// Insert buffer as unsecured data inside SignedData
// container
asn1cpp::setField (dataContentPayload->choice.
    unsecuredData, packetContent);
asn1cpp::setField (dataPayload->content,
    dataContentPayload);
asn1cpp::setField (signPayload->data, dataPayload);
asn1cpp::setField (tbs->payload, signPayload);

.....

asn1cpp::setField (ieeeData->content, ieeeContent);
// data encode
std::string encode_result = asn1cpp::oer::encode (
    ieeeData);
Ptr<Packet> packet = Create<Packet> ((uint8_t *)
    encode_result.c_str(), encode_result.size());

```

```

dataRequest.data = packet;
free (buffer);
return dataRequest;
}

```

Here, GNDataRequest\_t represents the GeoNetworking Data Request structure that is passed to the function for security enhancement.

### B.1.3 Certificate or Digest Selection Logic

This section outlines the logic to decide whether to send a certificate or just a digest based on the time elapsed since the last certificate was transmitted.

```

// For each second it will send a signer part with
// certificate, otherwise it will send digest.
if (Simulator::Now ().GetMilliseconds () -
    m_timestampLastCertificate >= 1000 ||
    m_timestampLastCertificate == 0)
    m_timestampLastCertificate = Simulator::Now ().
    GetMilliseconds ();

```

If more than a second has passed since the last certificate was sent, or if no certificate has been sent yet, the function triggers the transmission of the full certificate. Otherwise, it sends only a digest.

### B.1.4 ExtractSecurePacket

The process of extracting and verifying a secured packet is shown here. If the security is enabled and the Next Header (NH) field of the basic header indicates that the packet is secured (value 2), the function attempts to extract the secure data.

```

// 2) Check NH field
if(enableSecurity && basicHeader.GetNextHeader()==2)
{
    if(m_security->extractSecurePacket (
dataIndication) == Security::
SECURITY_VERIFICATION_FAILED) {
        discard_packet++;
    }
}

```

In this snippet, the function first checks if the NextHeader field indicates a secured packet (2). If enabled, the extractSecurePacket method verifies the

packet's signature. If the verification fails, the packet is discarded, and a counter is incremented.

### B.1.5 ExtractSecurePacket Function Definition

The extractSecurePacket function handles the decryption and signature verification of the packet to ensure its integrity and authenticity.

```
Security::Security_error_t Security::extractSecurePacket
    (GNDataIndication_t &dataIndication)
{
    // Create sequence of Ieee1609Dot2Data
    asn1cpp::Seq<Ieee1609Dot2Data> ieeeData_decoded;
    uint8_t *buffer;
    buffer = (uint8_t *) malloc ((dataIndication.data->
        GetSize ()) * sizeof (uint8_t));
    dataIndication.data->CopyData (buffer, dataIndication.
        data->GetSize ());
    std::string packetContent ((char *) buffer, (int)
        dataIndication.data->GetSize ());
    // Decode the packet
    ieeeData_decoded = asn1cpp::oer::decode (packetContent
        , Ieee1609Dot2Data);
    free (buffer);

    GNsecDP secureDataPacket;
    // Extract all fields to check signature and get
    // unsecured data
    secureDataPacket.protocol_version = asn1cpp::getField
        (ieeedata_decoded->protocolVersion, long);
    NS_LOG_INFO ("Ieee1609Dot2Data container, protocol
        version: " << secureDataPacket.protocol_version);
    // boolean value for getSeq, getSeqOpt
    bool getValue_ok;
    // content of Ieee1609Dot2Data
    auto contentDecoded = asn1cpp::getSeqOpt (
        ieeedata_decoded->content, Ieee1609Dot2Content, &
        getValue_ok);
    // check the present, here is always signed data
    auto present1 = asn1cpp::getField (contentDecoded->
        present, Ieee1609Dot2Content_PR);
    if (present1 == Ieee1609Dot2Content_PR_signedData)
```

```

{
    auto signedDataDecoded = asn1cpp::getSeqOpt (
        contentDecoded->choice.signedData, SignedData, &
        getValue_ok);

    ....

    // If validation is ok, return packet (unsecured data)
    Ptr<Packet> packet = Create<Packet> ((uint8_t *)
        secureDataPacket.content.signData.tbsData.
        unsecureData.c_str (), secureDataPacket.content.
        signData.tbsData.unsecureData.size ());
    dataIndication.data = packet;
    return SECURITY_OK;
}

```

This function takes a `GNDDataIndication_t` packet as input and returns a `Security_error_t` indicating whether the security verification passed or failed. The implementation verifies the signature and checks packet integrity.

**Security Verification Check:** This snippet shows how the signature validation is performed inside the `extractSecurePacket`.

```

if (m_receivedCertificates.empty()){
    NS_LOG_INFO("No certificate received");
    return SECURITY_VERIFICATION_FAILED;
}else {
    //for every item in map do signature
    verification
    bool signValid = false;
    for (auto const &item : m_receivedCertificates
) {
        if (signatureVerification(tbs_hex, item.
second.second,secureDataPacket.content.signData.
signature,item.second.first)) {
            signValid = true;
            break;
        }
        NS_LOG_ERROR("Signature verification
failed for current certificate");
    }
    if (!signValid) {

```



```

        return SECURITY_VERIFICATION_FAILED;
    }

```

## B.2 Signature Algorithm

### B.2.1 Cryptographic Functions for Signature

This section shows the helper functions that implement cryptographic operations, such as hashing, hash concatenation, and EC key pair generation, which are essential for securing the packet.

**SHA-256 Hash Function:** This function computes the SHA-256 hash for the input data, producing a fixed-size hash value.

```

void Security::computeSHA256 (const std::vector<unsigned
    char> &data, unsigned char hash[SHA256_DIGEST_LENGTH
    ])
{
    SHA256_CTX sha256;
    SHA256_Init (&sha256);
    SHA256_Update (&sha256, data.data (), data.size ());
    SHA256_Final (hash, &sha256);
}

```

The function takes input data as a `std::vector<unsigned char>` and computes its SHA-256 hash, which is stored in the hash array.

**Concatenation of Hashes:** The function concatenates multiple hash values, a typical step in constructing data for cryptographic signing.

```

std::vector<unsigned char> Security::concatenateHashes (
    const unsigned char hash1[SHA256_DIGEST_LENGTH],
    const unsigned char hash2[SHA256_DIGEST_LENGTH])
{
    std::vector<unsigned char> concatenatedHashes;
    concatenatedHashes.insert (concatenatedHashes.end (),
        hash1, hash1 + SHA256_DIGEST_LENGTH);
    concatenatedHashes.insert (concatenatedHashes.end (),
        hash2, hash2 + SHA256_DIGEST_LENGTH);
    return concatenatedHashes;
}

```

This snippet represents the concatenation of several hash values, which might be required before performing operations such as signing or verifying.

**EC Key Pair Generation:** The function generates an Elliptic Curve (EC) key pair, used in cryptographic operations such as signing and verification.

```
Security::GNPublicKey Security::generateECKeypair ()
{
    EC_KEY *ec_key = EC_KEY_new_by_curve_name (
        NID_X9_62_prime256v1);
    if (!ec_key)
    {
        std::cerr << "Error creating EC_KEY object" << std
        ::endl;
        print_openssl_error ();
        return {};
    }

    if (!EC_KEY_generate_key (ec_key))
    {
        std::cerr << "Error generating EC key pair" << std
        ::endl;
        print_openssl_error ();
        EC_KEY_free (ec_key);
        return{};
    }

    m_ecKey = EC_KEY_dup(ec_key);

    .....

    char *pub_key_hex = EC_POINT_point2hex (
        EC_KEY_get0_group (ec_key), pub_key_point,
        POINT_CONVERSION_COMPRESSED, ctx);
    if (!pub_key_hex)
    {
        std::cerr << "Error converting public key to hex"
        << std::endl;
        print_openssl_error ();
        BN_CTX_free (ctx);
    }
}
```

```

    EC_KEY_free (ec_key);
    return {};
}

// Remove prefix from the PK
std::string pub_key_hex_str (pub_key_hex);
std::string prefix = pub_key_hex_str.substr (0, 2);
if (prefix == "02")
    prefix = "compressed_y_0";
else if (prefix == "03")
    prefix = "compressed_y_1";

pub_key_hex_str = pub_key_hex_str.substr (2);

publicKey.prefix = prefix;
publicKey.pk = pub_key_hex_str;

EC_KEY_free(ec_key);

return publicKey;
}

```

The function creates an EC key pair, returning a public key for use in future cryptographic operations.

## B.2.2 Signing Hashes

This function is responsible for signing a hash using a private key, forming the core of the signature generation process.

```

// Function to sign a hash with a private key
ECDSA_SIG * Security::signHash (const unsigned char *
    hash, EC_KEY *ec_key)
{
    ECDSA_SIG *signature = ECDSA_do_sign (hash,
        SHA256_DIGEST_LENGTH, ec_key);
    if (!signature)
    {
        std::cerr << "Error signing hash" << std::endl;
        print_openssl_error ();
    }
    return signature;
}

```

---

}

---

The function `signHash` takes a hash and signs it with the EC private key. The result is an `ECDSA_SIG`, which can later be transmitted or stored as part of the secure packet.

## B.3 Signature Creation

This section describes how the signature is created for a packet. It typically involves hashing the To-Be-Signed (TBS) data and signing the result with the private key.

```
Security::GNsignMaterial Security::signatureCreation (
    const std::string& tbsData_hex, const std::string&
    certificate_hex)
{
    GNsignMaterial signMaterial;

    std::vector<unsigned char> tbsData_bytes =
        hexStringToBytes (tbsData_hex);
    std::vector<unsigned char> certificate_bytes =
        hexStringToBytes (certificate_hex);

    unsigned char tbsData_hash[SHA256_DIGEST_LENGTH];
    computeSHA256 (tbsData_bytes, tbsData_hash);

    unsigned char certificate_hash[SHA256_DIGEST_LENGTH];
    computeSHA256 (certificate_bytes, certificate_hash);

    std::vector<unsigned char> concatenatedHashes =
        concatenateHashes (tbsData_hash, certificate_hash)
        ;

    unsigned char final_hash[SHA256_DIGEST_LENGTH];
    computeSHA256 (concatenatedHashes, final_hash);

    EC_KEY *ec_key = EC_KEY_dup (m_ecKey);

    // Sign the final hash
    ECDSA_SIG *signature = signHash (final_hash, ec_key);
    if (!signature)
```

```

    {
        EC_KEY_free (ec_key);
    }

    ....

    signMaterial.r = r_padded_hex;
    signMaterial.s = s_padded_hex;

    // Clean up
    OPENSSL_free (r_hex);
    OPENSSL_free (s_hex);
    ECDSA_SIG_free (signature);
    EC_KEY_free (ec_key);

    return signMaterial;
}

```

This function receives the TBS data in hexadecimal format and a certificate in the same format. It generates a signature by computing the hash of the data and signing it using the private key associated with the certificate.

## B.4 Signature Verification

The signature verification process checks the authenticity of the packet by verifying that the received signature matches the expected value computed from the TBS data and the sender's public key.

```

bool Security::signatureVerification (const std::string&
    tbsData_hex, const std::string& certificate_hex,
    const GNsgtrDC& signatureRS, const std::string&
    verifyKeyIndicator)
{
    // Convert hex string to bytes
    std::vector<unsigned char> tbsData_bytes =
        hexStringToBytes (tbsData_hex);
    std::vector<unsigned char> certificate_bytes =
        hexStringToBytes (certificate_hex);

    // Compute SHA-256 hash

```

```
unsigned char tbsData_hash[SHA256_DIGEST_LENGTH];
computeSHA256 (tbsData_bytes, tbsData_hash);

unsigned char certificate_hash[SHA256_DIGEST_LENGTH];
computeSHA256 (certificate_bytes, certificate_hash);

// Concatenate the hashes
std::vector<unsigned char> concatenatedHashes =
    concatenateHashes (tbsData_hash, certificate_hash)
;

// Compute SHA-256 hash of the concatenated hashes
unsigned char final_hash[SHA256_DIGEST_LENGTH];
computeSHA256 (concatenatedHashes, final_hash);

.....

// Convert hex strings to bytes

std::vector<unsigned char> r_bytes(r_hex.begin(),
    r_hex.end());
std::vector<unsigned char> s_bytes(s_hex.begin(),
    s_hex.end());
std::vector<unsigned char> pk_bytes(verifyKeyIndicator
    .begin(), verifyKeyIndicator.end());

.....

// Setting signature through r and s values
if (!ECDSA_SIG_set0 (signature, r, s))
{
    std::cerr << "Error setting r and s in signature"
<< std::endl;
    print_openssl_error ();
    ECDSA_SIG_free (signature);
    EC_POINT_free (pub_key_point);
    EC_KEY_free (ec_key);
}

// Verify the signature
```

```
int verify_status = ECDSA_do_verify (final_hash,
    SHA256_DIGEST_LENGTH, signature, ec_key);
if (verify_status == 1)
{
    NS_LOG_INFO ("Signature is valid, received_hash ==
    computed_hash");
    validSignature = true;
}
else if (verify_status == 0)
{
    NS_LOG_INFO ("Signature is invalid");
    validSignature = false;
}
else
{
    std::cerr << "Error verifying signature" << std::
endl;
    print_openssl_error ();
}

// Clean up
ECDSA_SIG_free (signature);
EC_POINT_free (pub_key_point);
EC_KEY_free (ec_key);

return validSignature;
}
```

This function takes the TBS data, the certificate, the received signature, and a key indicator. It verifies that the signature corresponds to the hash of the TBS data using the public key extracted from the certificate, returning a boolean indicating success or failure.

## B.5 Encryption Algorithm for requests to Authorities

### B.5.1 Cryptographic Functions for Encryption

This section shows the helper functions that implement cryptographic operations, such as key derivation and message encryption, which are essential for ensuring the confidentiality and integrity of transmitted data.

**Key Derivation Using KDF2:** This function implements the KDF2 algorithm to derive cryptographic keys from a shared secret. The process iteratively hashes the secret with a counter and additional parameters until the desired key length is achieved.

```

void deriveKeyWithKDF2(const unsigned char* sharedSecret
    , size_t secretLen, const unsigned char* P1, size_t
    P1_len, unsigned char* derivedKey, size_t
    derivedKeyLen) {
size_t hBits = SHA256_DIGEST_LENGTH * 8; // SHA-256
    produces 256 bits size_t cThreshold = (derivedKeyLen
    * 8 + hBits - 1) / hBits;
size_t offset = 0;
unsigned int counter = 1;
std::vector<unsigned char> hashInput(secretLen + P1_len
    + 4);
std::memcpy(hashInput.data(), sharedSecret, secretLen);

while (offset < derivedKeyLen) {
    // Append counter (big endian) to the hash input
    hashInput[secretLen + 0] = (counter >> 24) & 0xFF;
    hashInput[secretLen + 1] = (counter >> 16) & 0xFF;
    hashInput[secretLen + 2] = (counter >> 8) & 0xFF;
    hashInput[secretLen + 3] = counter & 0xFF;

    // Append P1 to the hash input
    std::memcpy(hashInput.data() + secretLen + 4, P1,
    P1_len);

    // Compute hash
    unsigned char hash[SHA256_DIGEST_LENGTH];
    computeSHA256(hashInput, hash);

    // Copy hash output into derived key buffer
    size_t copyLen = std::min(derivedKeyLen - offset, (
    size_t)SHA256_DIGEST_LENGTH);
    std::memcpy(derivedKey + offset, hash, copyLen);
    offset += copyLen;
    counter++;
}
}

```



## B.5.2 Encryption Process

The encryption process secures the plaintext message using AES-CCM for confidentiality and integrity, combined with ephemeral ECDH key exchange to derive session keys. The following steps detail the process.

**AES Key Generation and Encryption:** An AES symmetric key is randomly generated and used to encrypt the plaintext with AES-CCM. A nonce is also generated to ensure the uniqueness of the encryption.

```
// Generate random AES key
unsigned char aesKey[AES_KEY_LENGTH];
if (RAND_bytes(aesKey, AES_KEY_LENGTH) != 1) {
    handleErrors(); }

// Generate nonce for AES-CCM
nonce.resize(NONCE_LENGTH);
if (RAND_bytes(nonce.data(), NONCE_LENGTH) != 1) {
    handleErrors(); }

// Initialize AES-CCM context
std::vector<unsigned char> ciphertext(plaintext.size());
EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
if (!ctx) handleErrors();
EVP_EncryptInit_ex(ctx, EVP_aes_128_ccm(), nullptr,
    nullptr, nullptr);
EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_CCM_SET_IVLEN,
    NONCE_LENGTH, nullptr);
EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_CCM_SET_TAG,
    AES_CCM_TAG_LENGTH, nullptr);
EVP_EncryptInit_ex(ctx, nullptr, nullptr, aesKey, nonce.
    data());

// Encrypt plaintext
int len;
EVP_EncryptUpdate(ctx, nullptr, &len, nullptr, plaintext
    .size());
EVP_EncryptUpdate(ctx, ciphertext.data(), &len,
    plaintext.data(), plaintext.size());
```

```

int ciphertext_len = len;
EVP_EncryptFinal_ex(ctx, ciphertext.data() + len, &len);
ciphertext_len += len;

// Retrieve AES-CCM tag
aesCcmTag.resize(AES_CCM_TAG_LENGTH);
EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_CCM_GET_TAG,
    AES_CCM_TAG_LENGTH, aesCcmTag.data());
EVP_CIPHER_CTX_free(ctx);
ciphertext.resize(ciphertext_len);

```

**Derivation of Shared Secret and Session Keys:** The shared secret is derived using ECDH between the ephemeral private key and the receiver's public key. Keys  $ke$  (for encryption) and  $km$  (for integrity) are then derived from the shared secret using KDF2.

```

// Generate ephemeral EC key pair
EVP_PKEY_CTX *pctx = EVP_PKEY_CTX_new_id(EVP_PKEY_EC,
    nullptr);
EVP_PKEY_keygen_init(pctx);
EVP_PKEY_CTX_set_ec_paramgen_curve_nid(pctx,
    NID_X9_62_prime256v1);
EVP_PKEY *ephemeralKey = nullptr;
EVP_PKEY_keygen(pctx, &ephemeralKey);
EVP_PKEY_CTX_free(pctx);

// Derive shared secret using ECDH
EVP_PKEY_CTX *deriveCtx = EVP_PKEY_CTX_new(ephemeralKey,
    nullptr);
EVP_PKEY_derive_init(deriveCtx);
EVP_PKEY_CTX_set_ecdh_cofactor_mode(deriveCtx, 1);
EVP_PKEY_derive_set_peer(deriveCtx, receiverPublicKey);

size_t secretLen;
EVP_PKEY_derive(deriveCtx, nullptr, &secretLen);
std::vector<unsigned char> sharedSecret(secretLen);
EVP_PKEY_derive(deriveCtx, sharedSecret.data(), &
    secretLen);
EVP_PKEY_CTX_free(deriveCtx);

// Derive ke and km

```

```
unsigned char derivedKey[48];
deriveKeyWithKDF2(sharedSecret.data(), secretLen, p1,
    SHA256_DIGEST_LENGTH, derivedKey, sizeof(derivedKey))
;

unsigned char ke[16], km[32];
memcpy(ke, derivedKey, 16); // ke is 16 bytes
memcpy(km, derivedKey + 16, 32); // km is 32 bytes
```

**Encryption of AES Key and Integrity Check:** The AES key is XOR-encrypted using ke, and an HMAC tag is generated using km to ensure message integrity.

```
// Encrypt AES key with XOR and ke
encryptedKey.resize(AES_KEY_LENGTH);
for (size_t i = 0; i < AES_KEY_LENGTH; i++) {
    encryptedKey[i] = aesKey[i] ^ ke[i];
}

// Compute HMAC for integrity using km
eciesTag.resize(16);
HMAC_CTX* hctx = HMAC_CTX_new();
HMAC_Init_ex(hctx, km, 32, EVP_sha256(), nullptr);
HMAC_Update(hctx, encryptedKey.data(), encryptedKey.size
    ());
unsigned int hmacLen = 0;
HMAC_Final(hctx, eciesTag.data(), &hmacLen);
HMAC_CTX_free(hctx);
```

# Bibliography

- [1] Yoshizawa Takahito, Dave Singelée, Jan Mühlberg, Delbruel Stéphane, Amir Taherkordi, Danny Hughes, and Bart Preneel. «A Survey of Security and Privacy Issues in V2X Communication Systems». In: *ACM Computing Surveys* 55 (Aug. 2022). DOI: 10.1145/3558052 (cit. on pp. 3, 8).
- [2] Ahmad Alalewi, Iyad Dayoub, and Soumaya Cherkaoui. «On 5G-V2X Use Cases and Enabling Technologies: A Comprehensive Survey». In: *IEEE Access* 9 (2021), pp. 107710–107737. DOI: 10.1109/ACCESS.2021.3100472 (cit. on p. 4).
- [3] Junsung Choi, Vuk Marojevic, Mina Labib, Siddharth Kabra, Jayanthi Rao, Sushanta Das, Jeffrey Reed, and Carl Dietrich. *Regulatory Options and Technical Challenges for the 5.9 GHz Spectrum: Survey and Analysis*. Oct. 2018 (cit. on p. 5).
- [4] Yoshizawa Takahito and Bart Preneel. «Survey of Security Aspect of V2X Standards and Related Issues». In: Oct. 2019, pp. 1–5. DOI: 10.1109/CSCN.2019.8931311 (cit. on pp. 5, 7).
- [5] Muhammad Sameer Sheikh, Jun Liang, and Wensong Wang. «A Survey of Security Services, Attacks, and Applications for Vehicular Ad Hoc Networks (VANETs)». In: *Sensors* 19.16 (2019). ISSN: 1424-8220. DOI: 10.3390/s19163589. URL: <https://www.mdpi.com/1424-8220/19/16/3589> (cit. on pp. 8, 14).
- [6] Kalid Rabeh, Mohamed Mahmoud, Terry Guo, and Mohamed Younis. «Cross-Layer Scheme for Detecting Large-scale Colluding Sybil Attack in VANETs». In: vol. 1. June 2015. DOI: 10.1109/ICC.2015.7249492 (cit. on p. 12).
- [7] Jyoti Grover, Deepak Kumar, Sargurunathan Mohan, Manoj Gaur, and Vijay Laxmi. «Performance Evaluation and Detection of Sybil Attacks in Vehicular Ad-Hoc Networks». In: July 2010, pp. 473–482. ISBN: 978-3-642-14477-6. DOI: 10.1007/978-3-642-14478-3\_47 (cit. on p. 13).

- [8] *ETSI EN 302 665 V1.1.1 (2010-09) - Intelligent Transport Systems (ITS); Communications Architecture*. Standard. European Telecommunications Standards Institute, 2010 (cit. on p. 19).
- [9] *ETSI TS 102 940 V2.1.1 (2021-07) - Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management; Release 2*. Technical Specification. European Telecommunications Standards Institute, 2021 (cit. on pp. 23, 33).
- [10] *ETSI EN 302 636-1 V1.2.1 (2014-02) - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*. Standard. European Telecommunications Standards Institute, 2014 (cit. on p. 24).
- [11] *ETSI EN 302 636-4-1 V1.4.1 (2019-11) - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality*. Standard. European Telecommunications Standards Institute, 2019 (cit. on p. 25).
- [12] *IEEE 1609.2-2022 - IEEE Standard for Wireless Access in Vehicular Environments—Security Services for Application and Management Messages*. Standard. Institute of Electrical and Electronics Engineers, 2022 (cit. on pp. 28, 29, 32, 37).
- [13] *IEEE 1609.2-2022 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Certificate Management Interfaces for End Entities*. Standard. Institute of Electrical and Electronics Engineers, 2022 (cit. on p. 35).
- [14] *ETSI TS 102 941 V2.2.1 (2022-11) - Intelligent Transport Systems (ITS); Security; Trust and Privacy Management; Release 2*. Technical Specification. European Telecommunications Standards Institute, 2022 (cit. on pp. 36, 37).
- [15] *ETSI TS 103 097 V2.1.1 (2021-10) - Intelligent Transport Systems (ITS); Security; Security header and certificate formats; Release 2*. Technical Specification. European Telecommunications Standards Institute, 2021 (cit. on pp. 42, 56).
- [16] Tomas Kulik, Brijesh Dongol, Peter Gorm Larsen, Hugo Daniel Macedo, Steve Schneider, Peter W. V. Tran-Jørgensen, and James Woodcock. «A Survey of Practical Formal Methods for Security». In: *Form. Asp. Comput.* 34.1 (2022). ISSN: 0934-5043. DOI: 10.1145/3522582. URL: <https://doi.org/10.1145/3522582> (cit. on p. 45).
- [17] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. «ProVerif 2.05: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial». In: (Oct. 2023) (cit. on p. 46).

- [18] F. Raviglione, C.M. Risma Carletti, M. Malinverno, C. Casetti, and C.F. Chiasserini. «ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services». In: *Computer Communications* 217 (2024), pp. 70–86. ISSN: 0140-3664. DOI: <https://doi.org/10.1016/j.comcom.2024.01.022>. URL: <https://www.sciencedirect.com/science/article/pii/S0140366424000227> (cit. on p. 53).
- [19] *ETSI TS 102 965 V2.2.1 (2024-07) - Intelligent Transport Systems (ITS); Application Object Identifier (ITS-AID); Registration; Release 2*. Technical Specification. European Telecommunications Standards Institute, 2024 (cit. on p. 56).
- [20] Marco Rapelli, Francesco Raviglione, and Claudio Casetti. «Oscar: An ETSI - Compliant C- ITS Stack for Field- Testing with Embedded Hardware Devices». In: *2024 22nd Mediterranean Communication and Computer Networking Conference (MedComNet)*. 2024, pp. 1–4. DOI: [10.1109/MedComNet62012.2024.10578207](https://doi.org/10.1109/MedComNet62012.2024.10578207) (cit. on p. 66).
- [21] Jiaqi Huang, Dongfeng Fang, Yi Qian, and Rose Qingyang Hu. «Recent Advances and Challenges in Security and Privacy for V2X Communications». In: *IEEE Open Journal of Vehicular Technology* 1 (2020), pp. 244–266. DOI: [10.1109/OJVT.2020.2999885](https://doi.org/10.1109/OJVT.2020.2999885) (cit. on p. 76).