D4.1
CARAMEL PKI enabled Vehicle Identity Management System

Topic	SU-ICT-01-2018
Project Title	Artificial Intelligence-based Cybersecurity for Connected and Automated Vehicles
Project Number	833611
Project Acronym	CARAMEL
Contractual Delivery Date	M16
Actual Delivery Date	M18
Contributing WP	WP4
Project Start Date	01/10/2019
Project Duration	30 Months
Dissemination Level	PU
Editor	ATOS
Contributors	I2CAT, ALTRAN, UBIWR

Document History		
Version	Date	Remarks
0.1	05/10/2020	Initial document structure, table of contents
0.2	21/01/2021	Added contribution to section 3
0.3	25/01/2021	Added contribution to section 4 and 4.2
0.4	28/01/2021	Added contribution to section 7
0.5	04/02/2021	Added contribution to section 4.3
0.6	26/02/2021	Added contribution to section 2
0.7	26/02/2021	Added contribution to section 5.2.3
0.7	03/03/2021	Added contribution to section 5.2.1
0.9	08/03/2021	Added contributions to section 5.2
1.0	18/03/2021	Initial version for internal and SAB review
1.1	26/03/2021	Updated to address the comments received from the internal review
1.2	31/03/2021	Final version

Contributors		
Name	**Organisation**	**Contribution**
Rodrigo Diaz	ATOS	Editor
Nicola Gregorio Durante	ATOS	Editor
Javier Ruiz Lozano	ATOS	Editor
Josep Escrig Escrig	I2CAT	Contributor
Sergi Sánchez	i2CAT	Contributor
Sergi Mercadé	i2CAT	Contributor
Kirti Kumar Saikia	ALTRAN	Contributor
João Serrano	UBIWR	Contributor
Gemma Roqueta	FICOSA	Reviewer

DISCLAIMER OF WARRANTIES

CARAMEL has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833611. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

DISCLOSURE STATEMENT

Table of Contents

List of Figures

List of Tables

List of Acronyms

AA	Authorization Authority
AML	Adversarial Machine Learning
ANSI	American National Standards Institute
AT	Authorization Ticket
API	Application Programming Interface
BC	Bootstrap Certificate
CA	Certification Authority
CAL	Certificate Access Lists
CAM	Certificate Access Manager
CRL	Certificate Revocation List
CTL	Certificate Trust list
DH	Diffie-Hellman
EA	Enrolment Authority
EC	Enrolment Certificate
ETSI	European Telecommunications Standards Institute
FPP	False Positive Probability
HSM	Hardware Security Module
IETF	Internet Engineering Task Force
IoT	Internet of Things
ISO	International Organization for Standardization
ITS	Intelligent Transport System
ITS-S	ITS Station
MitM	Man-in-the-middle
NIST	National Institute of Standards & Technology
OBU	On-Board Unit
OCA	Online Certification Authority
OCSP	Online Certificate Status Protocol
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PKIX	Public Key Infrastructure X.509
RA	Revocation Authority
RCA	Root Certification Authority
RSU	Road Side Unit
SAE	Society of Automotive Engineers
VANET	Vehicular ad-hoc networks

Executive Summary

The CARAMEL project aims to proactively address cybersecurity challenges in the context of cooperative, connected and automated mobility vehicles. In order to materialize and foster self-driving vehicles (SAE L4/L5) acceptance, some of the key enabling technologies such as the V2X communications must be secure, privacy preserving and reliable. Security and privacy have been already identified by the European Telecommunications Standards Institute (ETSI) as main concerns and widely addressed in several standards. The security architecture for Intelligent Traffic Systems (ITS) defined by ETSI is based on a vehicular Public Key Infrastructure (PKI) that delivers digital certificates to vehicles and Road Side Units (RSUs) with the objective of securing V2X message transmissions.

This document reports the work done in the context of Task 4.1 of CARAMEL project, which aims to define a PKI-based Identity Management System for Vehicles. The defined solution will be fully compliant with ETSI standards and addresses the specific security and privacy concerns mentioned before. Besides this, we also address in this document new challenges and conflicts between these security and privacy requirements that Vehicular Ad-hoc Networks (VATNETs) may bring. More specifically, we addressed anonymity and scalability that are fundamental requirements for the PKI adoption in VANETs. In this regard, D4.1 provides a detailed view on the current state of the art, the mechanisms to put in place to fully address the security and privacy challenges, the PKI Toolbox architecture, including the description of the main components, its interfaces and details about the prototype implementation and deployment.

1 Introduction

The CARAMEL project aims to proactively combat cybersecurity challenges in the context of cooperative, connected and automated mobility vehicles. In order to materialize and foster self-driving vehicles (SAE L4/L5) acceptance, some of the key enabling technologies such as the V2X communications must be secure, privacy preserving and reliable. Security and privacy have been already identified by the European Telecommunications Standards Institute (ETSI) as main concerns and widely addressed in several standards [1]. The security architecture for Intelligent Traffic Systems (ITS) defined in [2] is based on a vehicular Public Key Infrastructure (PKI) that delivers digital certificates to vehicles and Road Side Units (RSUs) with the objective of securing V2X message transmissions. For enhancing privacy, V2X standards recommend to use short-term identities and change them periodically to hinder the tracking of the identities. However, knowing the location of vehicles and the interval used in short-term identity renewal, the tracking by an attacker becomes trivial.

The work done in WP4, and more specifically in T4.1, aims to define a PKI-based Identity Management System for Vehicles, fully compliant with ETSI standards, and to address the security and privacy challenges identified before, as well as other issues such as the scalability of the system.

1.1 Purpose of this Document

The purpose of this document is to define the CARAMEL PKI-enabled Vehicle Identity Management System compliant with the project architecture described in the deliverable D2.4 "System Specifications and Architecture". More precisely, this report overviews various components, interface models, architectures, requirements, and the implementation details of the PKI.

1.2 Structure of this Document

The current document is organized as follows:

- Section 1: includes the overview of the document context and links with other activities inside the project.
- Section 2: introduces basic concepts required to understand the following chapters, provides an overview of the current state of the art and applicable standards in this field.
- Section 3: provides a high-level overview of the PKI Toolbox architecture and security functions.
- Section 4: describes in detail the security issues and challenges addressed when defining the PKI Toolbox architecture.
- Section 5: presents a detailed view of the PKI Toolbox architecture and design, including the description of the main components and its interfaces.
- Section 6: provides details about the prototype implementation and deployment.
- Section 7: highlights the innovations addressed in this activity and the progress beyond the state of the art.
- Section 8: concludes this document.

1.3 Relation to other Tasks and Deliverables

D4.1 is linked to the following tasks and deliverables:

- Task 2.1 – Use Cases Elaboration / D2.1 Report on Detailed Specification of Use Cases: D4.1 takes into consideration the CARAMEL use cases defined in D2.1, as well as follows the technical evaluation strategy deployed within the project.

- Task 2.3 – Analysis of Security and Privacy Requirements / D2.2 Report on Threat Analysis and Cyber-Threats / D2.3 Specifications of CARAMEL Security and Privacy Requirements: D4.1 considers the user, security and privacy requirements defined in D2.2 and D2.3, with focus on the requirements affecting to the security of the V2X communications.
- Task 2.4 – System Specifications and Architecture / D2.4 - System Specifications and Architecture: D4.1 uses as basis the CARAMEL architecture defined in D2.4 and the initial definition of the PKI Toolbox included in this deliverable.
- Task 3.3 – Cyberthreat Detection and Response Techniques for Cooperative Automated Vehicles / D3.6 Cyberthreat Detection and Response Techniques for Cooperative Automated Vehicles: D4.1 complements the definition of the PKI core entities already introduced in D3.6.
- Task 5.1 – Collection and Storage of Data from Smart Vehicle's Internal Network / D5.2 Report on the Collection and Storage of Data from Smart Vehicle's Internal Network: D4.1 uses the CARAMEL V2X dataset described in D5.2 for training the Random Forest model in Section 4.4.

2 State of the Art of PKI Architectures and Standards

2.1 Introduction

Encryption techniques have been in use for more than a millennium to protect critical communications, like the clay tablets from Mesopotamia [3]. It is one of the oldest forms of science known to humans and showcases the desire to protect by disguising sensitive information to something unknown and/or obscure. There could be several possibilities of how encryption might have started, however it is known to be first started in Ancient Egypt [4]. And as the time changed the complexities of encryption techniques have evolved from Skytale, Caesar Cipher to Enigma, to the modern encryption standards like Diffe-Hellman, RSA, DES, etc. [4]. Modern day Cryptography is the technique or method to protect sensitive information and communication by the means of codes, as the name goes ‘Crypt’ which means ‘hidden’ or ‘vault’ and ‘graphy’ means ‘writing’ i.e., ‘Hidden Writing’ [5]. Cryptography supports in attaining the below mentioned services to secure information and communication in storage or during transit [5]:

1. **Confidentiality:** Sensitive information is accessible to authorized personnel only.
2. **Integrity:** Ensuring accuracy and completeness of sensitive information during its entire lifecycle.
3. **Authentication:** Verification of sender and receiver identity for sensitive information.
4. **Non-repudiation:** A sender of sensitive information cannot deny creation or transmission of the information.

There are 2 types of cryptography [3]:

1. **Symmetric Cryptography:** It is the most well-known encryption technique which utilizes a single key to encrypt and decrypt information. It is very fast & efficient, however provides only ‘Confidentiality’ security service as other services are not achievable with a single key. It is also known as Private Key, Secret Key, Shared Key or Session Key Cryptography.
2. **Asymmetric Cryptography:** It utilizes a pair of keys, Private Key which is only available with the user as a secret key and Public Key which is shared with everyone else. It is complex & slow, however overcomes most of the challenges faced by Symmetric Cryptography. It provides ‘Confidentiality’, ‘Authenticity’ and ‘Non-repudiation’ security services.

Apart from its speed, a crucial challenge faced by asymmetric cryptography is the mapping of public keys to the receiver of information, which makes it prone to Man-in-the-middle (MitM) attacks. This issue was eventually solved by the introduction of Public Key Infrastructure (PKI). PKI is a set of roles, policies and procedures needed to create, store, use, distribute, manage and revoke digital certificates necessary to manage public key encryption [5]. PKI emerged in the 1990s to govern issuance, revocation and management of encryption keys through Digital certificates [6]. The 1st wave of PKI (1995-2002) was mostly focused on eCommerce Websites to provide consumers with the assurance of visiting the right website and security for financial transactions. The 2nd wave (2003-2010) was the emergence of Enterprise PKI for authenticating and verifying their Mobile workforce, and finally the 3rd wave (2011 – till date) which covers the earlier areas with new use cases like multi-device workforce, fully mobile, etc. and IoT as well as new developments around Autonomous and Connected vehicle requirements [6]. In present day with services around Web and Mobile Banking, Internet of Things (IoT) management and updates, Autonomous and Connected vehicles, Cloud Computing, Blockchain, etc., PKI has become a de-facto technology to ensure security of information in transmission as well as in storage.

The main function of PKI is to manage encryption keys and digital certificates as well as to perform the below mentioned task [7]:

1. Provide trust
2. Certificate generation
3. Certificate revocation
4. Certificate & Certificate Revocation list (CRL) storage
5. Key management

The PKI utilizes the trust model defined in the X.509 standard, as shown in Figure 1 below [7]:

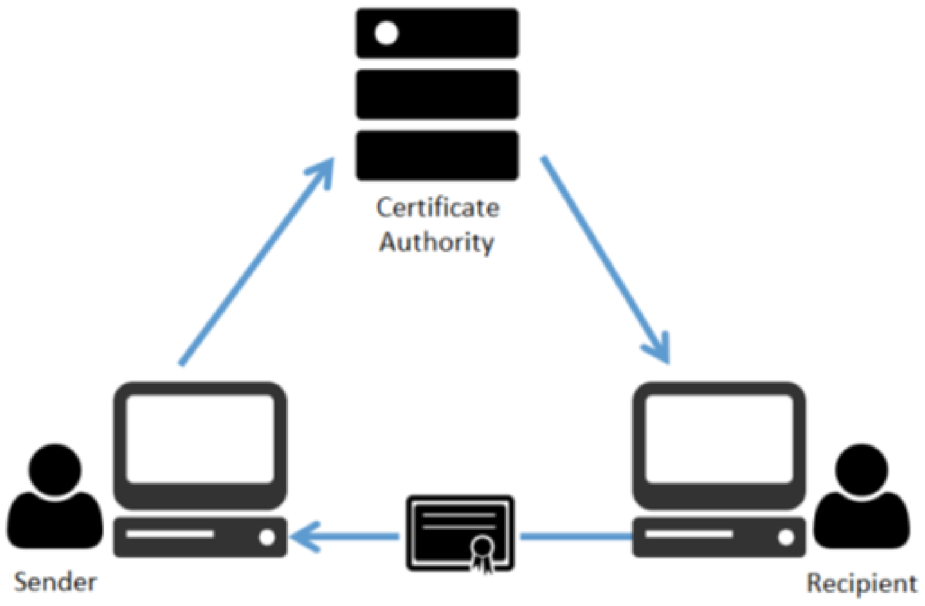

Figure 1: X.509 Trust Model

The key entities of a PKI are discussed below and achievement of these functions can be through a procedure or a technology and in some cases by the mixture of both [8][9]:

1. **Public & Private Key pair:** The basic requirement of PKI utilizing an associated pair of keys for encryption and decryption.
2. **Digital Certificates:** It is a digital document signed by the CA to associate the public-private key pair with the identity of the owner. The latest widely used certificate formats are all based on X.509v3 (RFC 5280) [10].
3. **Certificate Authority (CA):** Its main task is to issue as well as renew digital certificates and acts as a trust component, as any certificate issued by the CA is trusted by all users that trust the specific CA.
4. **Registration Authority (RA):** It receives certificate requests and verifies the identity credential provided by the user. Once approved, it initiates the certification process.
5. **Validation Authority (VA):** It allows a user to check the status of a certificate with regard to validity period as well as revocation status. It often utilizes OCSP (Online Certificate Status Protocol) or CRL (Certificate Revocation List) to advertise information about revoked certificates.
6. **Secure Storage:** A method to ensure security of private keys while in storage by a CA or a user.

2.2 PKI Architectures

Today's business environment is global and capable of providing services across the world with the help of Internet resources. However, cybersecurity is one of the major risks that reduces the adoptability of E-business by different companies [11]. PKI is one of most widely used security control for a distributed environment in which e-businesses operate. Still, this distributed environment has their own set of challenges while implementing a PKI architecture due to involvement of multiple CA's, trust relationship between the CA's, etc. There are multiple PKI architectures available, however they can be classified in one of the fundamental architectures provided below [11]:

1. **Simple PKI Architectures:**
    a. Single CA Architecture: This architecture consists of only one CA that provides services to all entities in the PKI environment and entities trust only one CA (Figure 2). This architecture is suitable for small businesses with limited users.

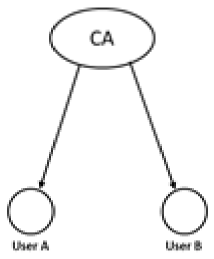

Figure 2: Single CA Architecture

    b. Basic Trust List Architecture: This architecture establishes trust relationships with CA's based on a 'Trust list' that is available for the users (Figure 3). Trust list can be a certificate list (like web browser certificate storage) or a signed list containing confidential information (like certificate hash or file names or Certificate Trust list (CTL) for Microsoft certificate).

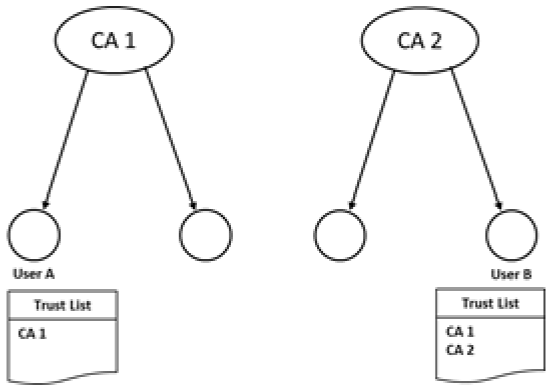

Figure 3: Basic Trust List Architecture

2. **Enterprise PKI Architecture:**
    a. Hierarchical PKI Architecture: This architecture follows a hierarchical structure as the name mentions and it is implemented by creating trust relationships between the root and the intermediate CA's (Figure 4). The root CA issues certificates only to intermediate CA's. However, the intermediate CA's issue certificates to other intermediate CA's and / or end users.

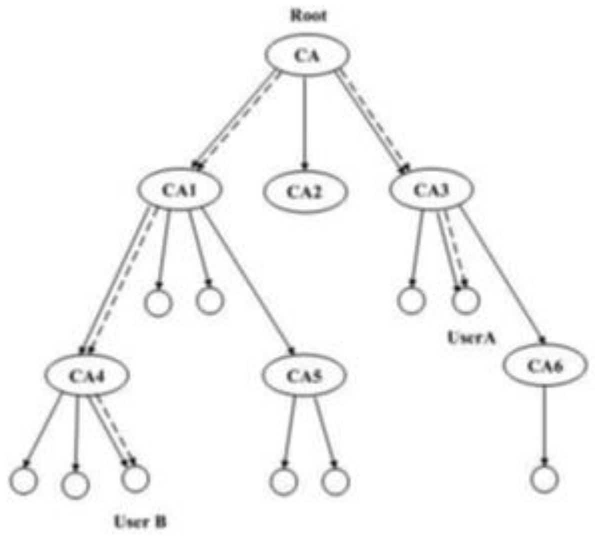

Figure 4: Hierarchical PKI Architecture [11]

b. Mesh PKI Architecture: This architecture establishes peer-to-peer bi-directional trust relationships between equal CA's, as shown on Figure 5. CA's might restrict trust by issuing certificates containing restrictions, like name, policy or path-length constraints.

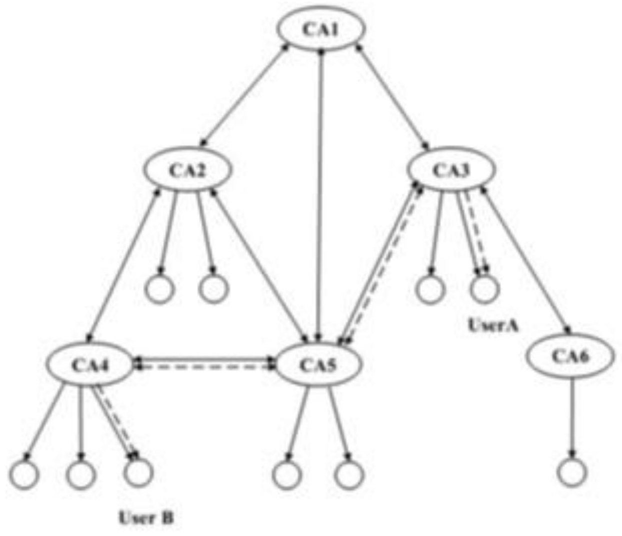

Figure 5: Mesh PKI Architecture [11]

3. Hybrid PKI Architecture:

a. Extended Trust List Architecture: This architecture establishes trust relationships by sustenance of trust list by the end users. Trust can be established in both hierarchical and mesh models, as entries in the trust list would contain the root CA from the hierarchy and few CA's from the mesh architecture (Figure 6).

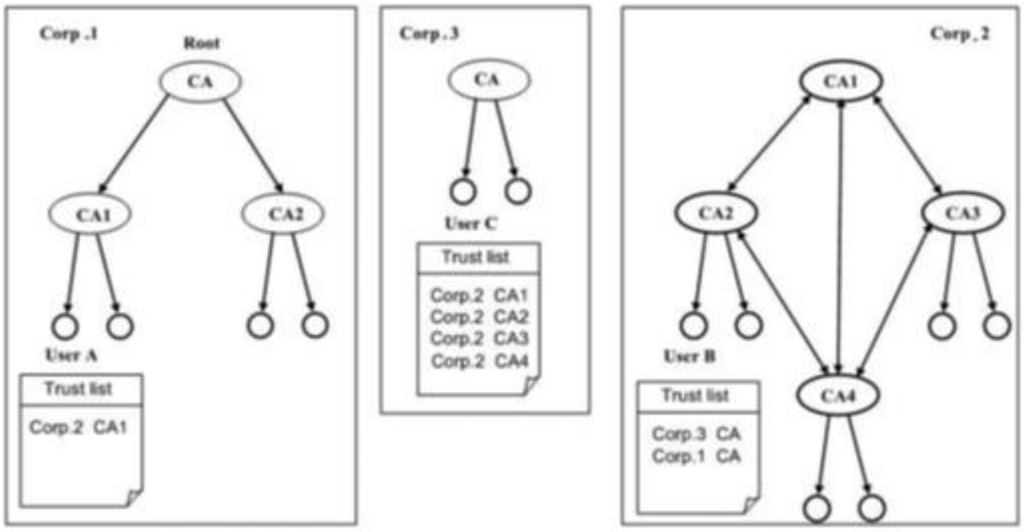

Figure 6: Extended Trust List Architecture [11]

b. Cross-Certified Enterprise PKI: This architecture establishes a peer-to-peer trust relationship with cross-certification and can be restricted by defining restrictions in one or more cross-certificate pair extensions. In this architecture, the root or intermediate CA of a hierarchical architecture can establish a peer-to-peer trust relationship with any CA from single CA, or CA's from mesh or other hierarchical architecture and in similar way, a single CA or mesh CA can establish trust relationships with other or similar architectures, as shown in Figure 7 with the double lines between Enterprise PKI Architectures.

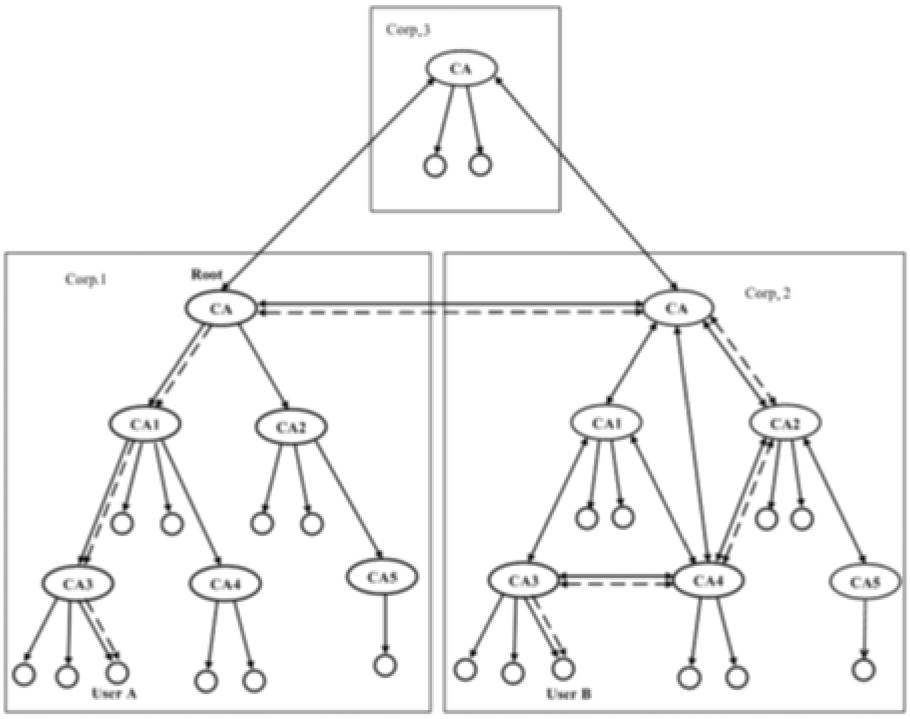

Figure 7: Cross-Certified Enterprise PKI Architecture [11]

c. Bridge Certification Authority Architecture: This architecture connects different PKI architectures by introducing a New Bridge CA to establish relations. The Bridge CA is not a trust point or issues any digital certificates. In fact, it is considered as a mediator between different PKI architectures.

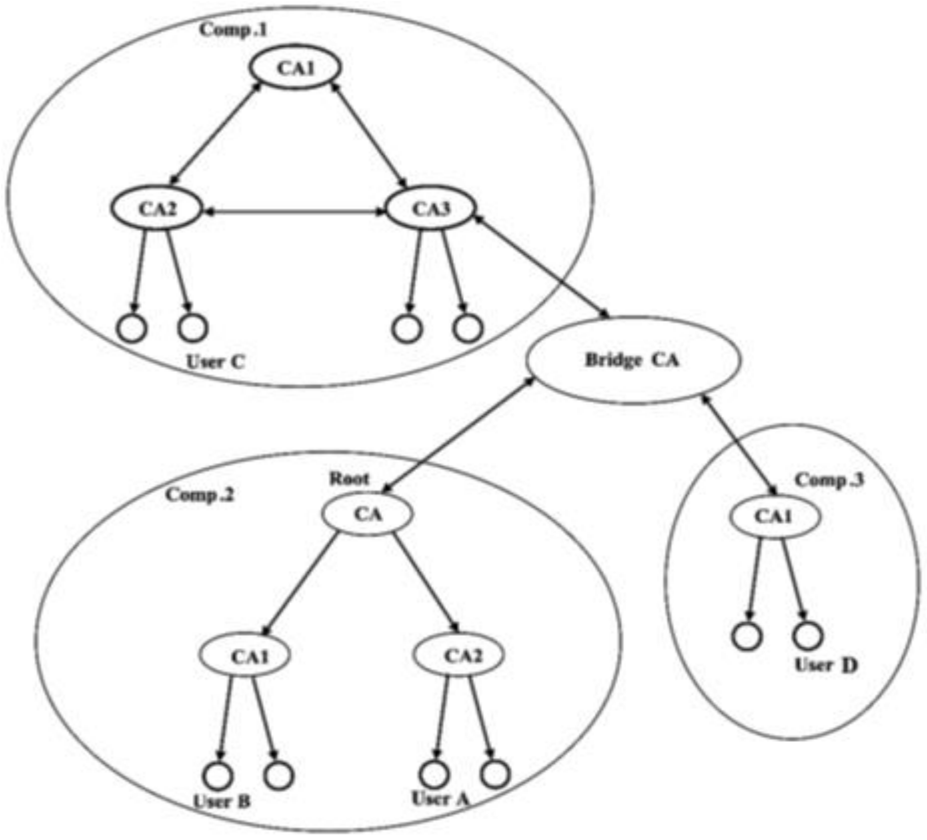

Figure 8: Bridge PKI Architecture [11]

All above mentioned architectures have their own advantages and challenges. However, identifying the best architecture would depend on a variety of parameters of the business including the requirement, scalability, flexibility, trust points, trust relationships, etc. The 2 most common architectures of PKI being utilized in the present-day are discussed below [12]:

1. **Pretty Good Privacy (PGP):** PGP was designed by Phil Zimmermann in 1991. It utilizes a 'Direct Trust' model and implements a concept of 'Web of Trust', where the users generate their own pair of keys and are published to other users for signing the key to become a 'Introducer' of the key. In this architecture, any user can act as a CA and validate other user's public keys. However, the certificate is valid for other users, only if they have identified the validator as a 'Trusted Introducer' [12] [13].

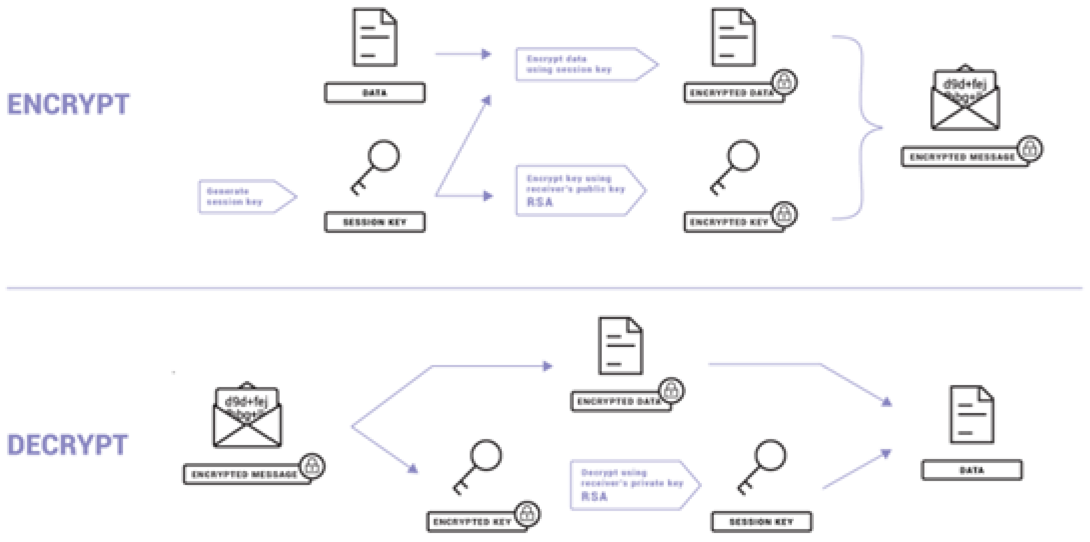

Figure 9: PGP Encryption [14]

2. **Internet X.509 Public Key Infrastructure (PKIX):** Since 1995, the Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) was established and have been working actively to setup a formal as well as generic model based on X.509 for an appropriate certificate-based architecture to be deployed on the Internet [12] [15]. This architecture utilizes a 'Hierarchical Trust Model' and utilizes a CA to issue digital certificates to distribute the 'Public Keys' among the end users holding 'Private Keys'.

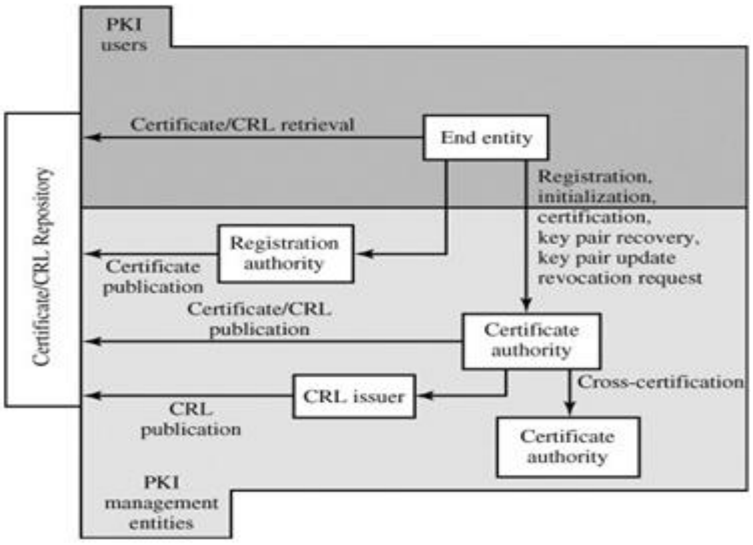
PKI users
Certificate/CRL Repository
Certificate/CRL retrieval
End entity
Registration, initialization, certification, key pair recovery, key pair update revocation request
Registration authority
Certificate publication
Certificate/CRL publication
Certificate authority
CRL issuer
CRL publication
Cross-certification
Certificate authority
PKI management entities

Figure 10: PKIX Architecture [15]

2.3 PKI Standards

There are a lot of standards for PKI and cryptography defined by different standardization organizations across the world. The list below provides information about some PKI-related standards defined by some well-known standardization organizations:

1. **Public Key Cryptography Standards (PKCS):** It is a set of standard protocols developed by RSA in 1990 for secure data exchange through PKI. The following is the list of standards that have been published [16][17]:

   a. PKCS #1 - RSA Cryptography Standard: Provides standards for implementation of RSA algorithm-based public key encryption and digital signature schemes.

   b. PKCS #3 - Diffie-Hellman Key Agreement Standard: Provides a method for implementation of Diffie-Hellman (DH) key agreement to share secret key between 2 parties.

   c. PKCS #5 - Password-Based Cryptography Standard: Provides a mechanism to achieve enhanced security for password-based cryptographic primitives.

   d. PKCS #6 - Extended-Certificate Syntax Standard: Defines extensions for X.509 v1 certificate specification. Also introduces X.509 v3 and moves the status of the standard to obsolete.

   e. PKCS #7 - Cryptographic Message Syntax Standard: Defines syntax used to digitally sign, digest, authenticate, or encrypt arbitrary message content. Also introduce IETF RFC 3369 which supersedes the PKCS #7 standard.

   f. PKCS #8 - Private-Key Information Syntax Standard: Defines syntax for private-key information including private key for some public key algorithms as well as a set of attributes for encrypted private key information.
   g. PKCS #9 - Selected Object Classes and Attribute Types: Defines two auxiliary object classes (pkcsEntity & naturalPerson) and some new attribute types & matching rules for other PKCS standards (e.g. PKCS #10 & PKCS #7).
   h. PKCS #10 - Certification Request Syntax Standard: Specifies syntax for public key certificate request.
   i. PKCS #11 - Cryptographic Token Interface Standard: Specifies an Application Programming Interface (API) known as 'Cryptoki' to devices which hold cryptographic information and perform cryptographic functions.
   j. PKCS #12 - Personal Information Exchange Syntax Standard: Describes a transfer syntax for personal identity information, including private keys, certificates, miscellaneous secrets and extensions. Introduction of IETF RFC 7292 supersedes the PKCS #12 standard [7].
   k. PKCS #15 - Cryptographic Token Information Syntax Standard: Defines a standard to enable components to be platform neutral, applications to be vendor neutral and usage of advanced technology for making applications neutral. It also specifies files and directories for storage of security-related information on cryptographic tokens.

2. **PKI X.509 (PKIX):** As already mentioned earlier, PKIX working group was established to develop X.509 based PKI standards. These standards form the basis of S/MIME (Secure Email), TLS (Secure TCP connections) and IPsec (Secure Internet Transactions) [7]. Few important PKI related RFCs are provided below [16]:
   a. RFC 3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile [18]
   b. RFC 4210: Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP) [19]
   c. RFC 6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP [20]
   d. RFC 3647: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework [21]
   e. RFC 4211: Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF) [22]
   f. RFC 5272: Certificate Management over CMS (CMC) [23]
   g. RFC 3739: Internet X.509 Public Key Infrastructure: Qualified Certificates Profile [24]
   h. RFC 3161: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP) [25]
   i. RFC 5755: An Internet Attribute Certificate Profile for Authorization [26]
3. **National Institute of Standards & Technology (NIST):**
   a. Special Publication (SP) 800-32: Introduction to Public Key Technology and the Federal PKI Infrastructure [27]
   b. Special Publication (SP) 800-15: MISPC Minimum Interoperability Specification for PKI Components, Version 1 [28]
4. **European Telecommunications Standards Institute (ETSI):**
   a. ETSI EN 319 411-1 V1.2.2 (2018-04): Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates; Part 1: General requirements [29]
   b. ETSI EN 302 665 V1.1.1 (2010-09): Intelligent Transport Systems (ITS); Communications Architecture [30]
   c. ETSI TS 103 097 V1.3.1 (2017-10): Intelligent Transport Systems (ITS); Security; Security header and certificate formats [31]

   d. ETSI TS 102 940 V1.3.1 (2018-04): Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management [2]
   e. ETSI TS 102 941 V1.3.1 (2019-02): Intelligent Transport Systems (ITS); Security; Trust and Privacy Management [1]

5. **American National Standards Institute (ANSI) [16]:**
   a. ANSI X9.79:2001: Financial Services Public Key Infrastructure (PKI) Policy and Practices Framework
   b. ANSI X9.31:1998: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)
   c. ANSI X9.57:1997: Public Key Cryptography for the Financial Services Industry: Certificate Management

6. **International Organization for Standardization (ISO):**
   a. ISO 21188:2018: Public key infrastructure for financial services — Practices and policy framework [32]
   b. ISO 17090-1:2013: Health informatics — Public key infrastructure — Part 1: Overview of digital certificate services [33]
   c. ISO/TR 21186-3:2021: Cooperative intelligent transport systems (C-ITS) — Guidelines on the usage of standards — Part 3: Security [34]
   d. ISO/IEC 9594-8:2020: Information technology — Open systems interconnection — Part 8: The Directory: Public-key and attribute certificate frameworks [35]

3 PKI Toolbox Overview

VANETs allow the exchange of road hazard warnings and information about the traffic, enabling automated reactions to overcome those issues. Despite the potential benefits introduced by this exchange of information, the associated wireless communication raises also security and privacy concerns that, if they are not properly addressed, could jeopardize their deployment because of the potential negative impact they could have on safety. To address the above commented concerns, the CARAMEL project aims to use a PKI-enabled Vehicle Identity Management System seeking to facilitate management of security credentials and the secure electronic transfer of information.

The general security requirements satisfied by PKI are authentication, confidentiality, integrity, availability and non-repudiation. In the PKI infrastructure, vehicles and messages are identified through authentication, ensuring that allowed vehicles are identified and that their messages are to be trusted. Confidentiality guarantees that the information contained in the transmitted messages is not made available or disclosed to unauthorized users. Integrity protects against the unauthorized creation, destruction or alteration of data. It ensures that a message was not altered or deleted during the transmission by a malicious node. Availability guarantees that the information, such as the revoked status of a vehicle, is available when needed. Non-repudiation ensures that users and entities are able to trace the origin of a message or action realized in the network if necessary.

Considering the specific case of the VANET, additional security requirements have been selected: anonymity and scalability. Anonymity ensures that the original identity of vehicles is not disclosed allowing vehicles not to be tracked. Any identity management system must consider to follow a privacy preserving scheme to protect vehicles and user's identity according to national and international legislation. Scalability is the ability of a system to handle a growing amount of work by adding resources to the system. Considering an ever-growing environment of connected vehicles, it is crucial to design a scalable and flexible system, as a progressive deployment is envisaged over time including multiple trust domains.

3.1 Background

Figure 11 illustrates the ITS-S Communication Reference Architecture defined by ETSI in [30]. It consists of four horizontal layers (the communication stack) and two cross layers used for management and security purposes.

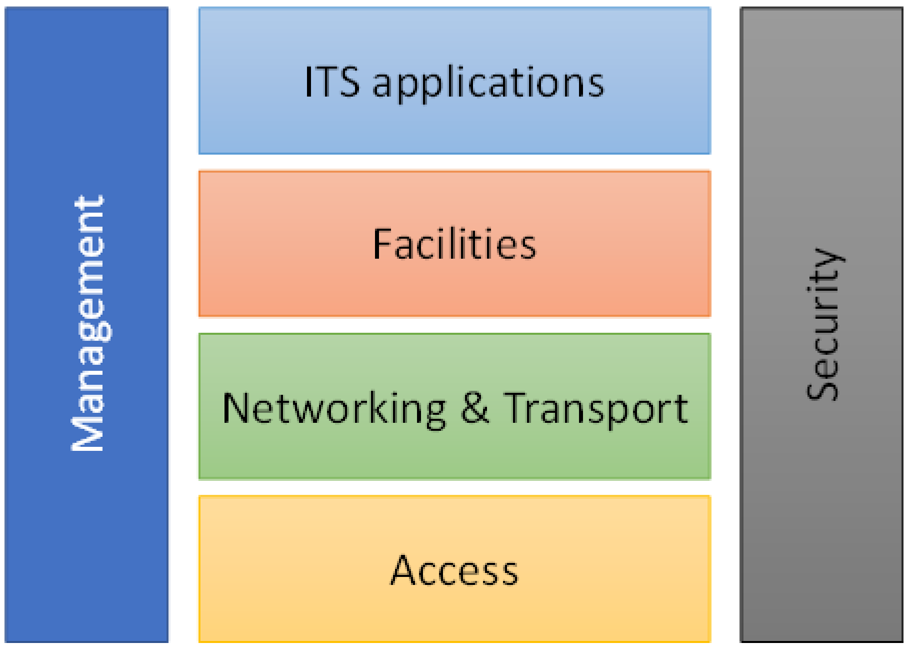

Figure 11: ETSI ITS-S reference architecture (from ETSI EN 302 665 [30])

This layered architecture has been further refined in the ETSI Technical Specifications for ITS communications security architecture and security management [31], where the security services are provided on a layer-by-layer basis. This way, each of the security services operates within one or several ITS architectural layers or within the Security Management layer.

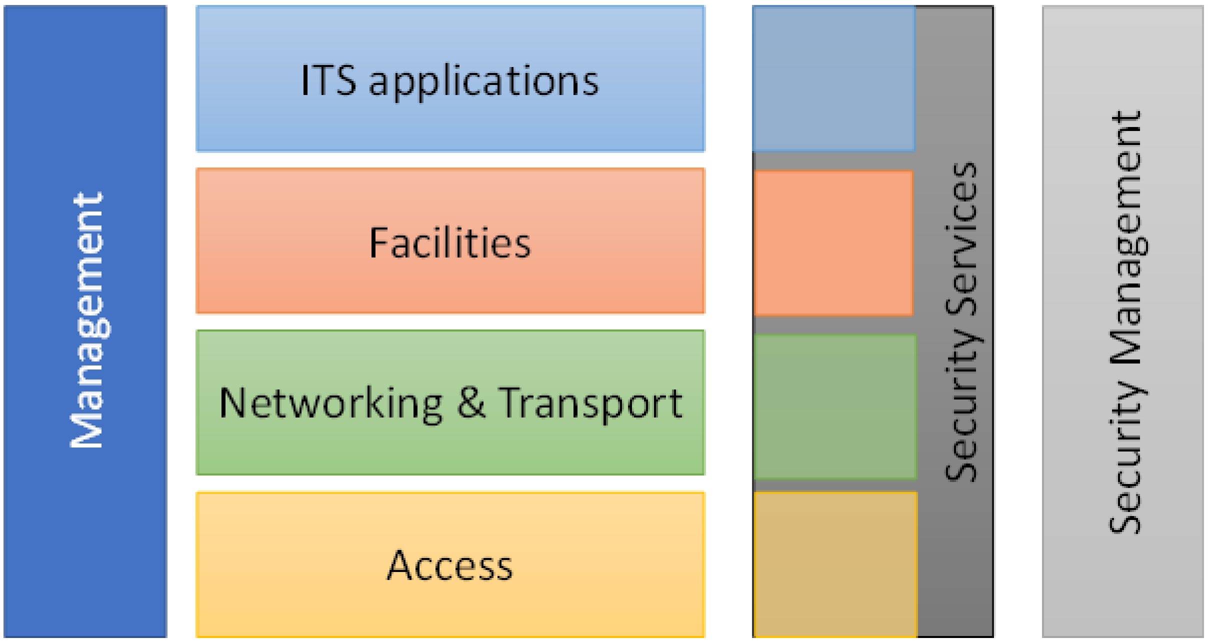

Figure 12: ETSI ITS-S security architecture (from ETSI TS 102 731 [31])

Both, security services and Security Management are in some way relying on a PKI architecture for ensuring the security of the communications

For defining the architecture of the CARAMEL PKI Toolbox, we have used as a starting point the PKI architecture defined by ETSI in [31], which includes the Enrolment and Authorization Authorities (AA), and we have extended it in the context of T4.1 activity, for addressing the revocation lifecycle not covered in the ETSI standard. For coordinating these extra functionalities, we have added a new authority called Revocation Authority (RA), which is in charge of the optimized management and distribution of the revoked certificates.

3.2 High-level Architecture

Figure 13 provides a high-level overview of the CARAMEL PKI-Toolbox architecture, which is composed by the following authorities, each satisfying a specific task:

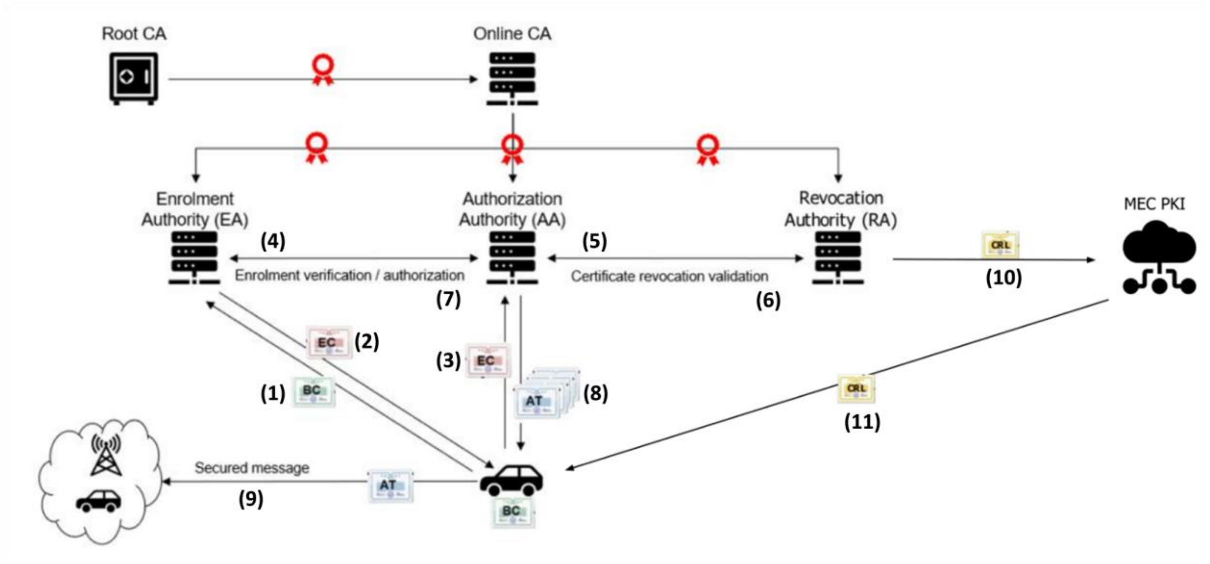
Root CA
Online CA
Enrolment Authority (EA)
Authorization Authority (AA)
Revocation Authority (RA)
MEC PKI
(4)
Enrolment verification / authorization
(5)
Certificate revocation validation
(7)
(6)
(10)
CRL
EC
(2)
(3)
(1)
BC
AT
(8)
(11)
Secured message
(9)

Figure 13: CARAMEL PKI-Toolbox High Level Architecture

- The Root Certification Authority (RCA) is an offline server that contains the root certificates for the entire PKI infrastructure. The server is offline in order to guarantee additional security and is in charge of signing the certificate belonging to the Online Certification Authority (OCA).
- The Online Certification Authority (OCA) is an online server signed by the RCA. The main responsibility of this authority is to sign the different lower authorities belonging to the PKI infrastructure.
- The Enrolment Authority (EA) is in charge of providing the necessary Enrolment Certificates (EC) at the enrolment phase. The ECs are used by the car to obtain the Authorization Tickets (AT) from the Authorization Authority (AA). An EC provides the long-term identity for the vehicle using the Bootstrap Credentials (BC), and allows it to obtain the ATs used to transmit the messages.
- The Authorization Authority (AA) is the entity which manages the Authorization Tickets (AT), also known as pseudonym certificates. ATs are issued to ensure privacy and anonymity of the vehicles within the PKI infrastructure. ATs provide the short-term identities to the vehicles and are the certificates used to encrypt and sign the messages transmitted by the vehicle.
- The Revocation Authority (RA) processes the revocation requests from the MEC PKI and generates the Certificate Revocation List (CRL) containing the revoked ATs. Moreover, the RA offers an internal API to check the validity of a specific AT in real-time. This functionality is used by the EA and the AA when a request to issue a new certificate (EC or ATs) is received.

3.3 Information Flow

The information flows illustrated by Figure 13 is described as follows:

(1) At the beginning of the lifetime of a vehicle, its manufacturer must request the registration of the vehicle through the Bootstrap Certificate (BC). Then the vehicle must perform an Enrolment Credential request by sending to the Enrolment Authority its BC. The EA verifies the request and in case of positive verification the EA registers the vehicle information in its database. This operation can be performed only once and only one EC for each vehicle can be issued.

(2) Once the EA receives the enrolment request, it authenticates and verifies that the information in the EC certificate request is valid for the associated vehicle. In case of positive validation, the EA issues the Enrolment Certificate (EC) and transmits it to the vehicle using an EC response message.

(3) Once the vehicle obtains its EC, it can perform an AT request to the AA. The AA communicates with the EA and the RA to assess the validity of the request (flow 4).

(4) The EA authenticates the requesting vehicle and validates whether the vehicle is entitled to get the requested AT according to its certificate policy. As part of this policy check, several properties of the EC are checked such as the revocation status (flows 5 and 6), time/region validity, permissions, assurance level, …

(5) The EA queries the revocation status of the received EC to the RA.

(6) The RA replies to the query with the revocation status.

(7) The EA returns a validation response once the policy has been validated.

(8) The AA generates the requested certificates (ATs) if the responses from the EA and the RA are positive and transmits them to the vehicle. If the AT request is not correct, the EA validation response is negative or the RA revocation response is negative, then AA refuses the AT request. If the vehicle still wants to obtain ATs, it can perform a new authorization request.

(9) With the obtained ATs, the vehicle is able to exchange V2X messages in a secure and anonymous way.

(10) The RA forwards to the MEC Revocation Services the Certificate Revocation List (CRL).

(11) The Revocation Services located in the MEC disseminate the CRL to the vehicles which use them to discard the received messages originating from revoked entities.

3.4 Security Functions

The CARAMEL PKI-enabled Vehicle Identity Management System has been designed to achieve the identified security requirements of confidentiality, integrity, availability, non-repudiation, anonymity and scalability.

The authentication and the consequent verification of the identity of the car is performed by means of the digital certificates. The message receiver can check the legitimacy of the sender by verifying the attached certificate. In our case, the certificates are assigned at two different levels. First, the Enrolment Authority assigns to a vehicle its long-term EC, from which the short-lived ATs are generated. Those short-lived certificates aim to conceal the real identity of the sender. The traceability is guaranteed by the mapping relationships of permanent identity and pseudonyms, which are stored in the CA. The EC uniquely identifies the vehicle and the ATs are used by the vehicle to sign and encrypt the messages it sends, protecting its anonymity.

The confidentiality, intended to prevent the unauthorized disclosure of information, is ensured through the use of encryption algorithms. The use of PKI guarantees that only an intended recipient can decrypt an encrypted message.

The integrity requirement is achieved using the short-term certificates, the ATs, to sign the messages. An altered message would produce a signature mismatch between the message and the attached signature and would cause the receiving vehicle to drop it. Only vehicles with a valid AT generated from the PKI are able to produce valid signatures for the messages, and on the other hand, the untrusted vehicles are identified and indicated to the active vehicles using Certificate Revocation Lists (CRL). CRLs are periodically distributed to the fleet of active vehicles to periodically update the vehicles on the untrusted actors which might contact them. Upon the reception of a message, if the certificate used to sign it is contained in the CRL, the received message is dropped.

Availability is addressed with the integration of the PKI with the MEC and the RSUs. The availability of an updated CRL is crucial for the vehicles, as failing to obtain an up-to-date list could lead to the acceptance of messages from revoked vehicles. To guarantee availability, vehicles are equipped with a high level of connectivity.

Non-repudiation is guaranteed requiring that all communications use encrypted and signed messages. In this manner, the PKI is able to trace the authorization tickets used to sign messages by the vehicles to the long-term identity of the vehicle itself. Moreover, PKI and RSU communication uses signed messages with the public key certificates assigned to them, guaranteeing non-repudiation.

Anonymity is ensured using the short-lived ATs instead of the long-term EC to sign the messages. Since only the PKI is able to derive the EC from an ATs, the short-lived certificates protect the real identity of the vehicles. As a result, vehicles are not able to distinguish two messages originating from the same vehicle and two messages sent from two different vehicles.

Scalability is a key element when considering the deployment of a PKI system, as the number of vehicles and consequently the EC and AT associated to them, can grow rapidly. To address this, the PKI uses federations which divide the fleet of vehicles into geographical regions, scaling down the number of vehicles to be accounted for. Since the number of ATs for each vehicle is quite high, with the growing number of revoked vehicles over time, the size of a CRL can become excessive. The usage of federations dividing the revoked vehicles into geographical regions and compressing the CRL using space efficient bloom filters keep the CRL size under control.

4 Security Issues and Challenges

As we introduced in the previous section, anonymity and scalability are fundamental requirements in the framework of PKI for VANETs. Privacy and auditability should be delicately balanced to not limit user acceptance. In this regard, the real identity of a driver/vehicle shouldn't be exposed under no circumstance to any other VATNET entity. Nevertheless, as in many other communication networks, it may be the case that a law-enforcement agency needs to trace the identity of the vehicles present in a concrete location in the course of an incident investigation. These apparent contradictory requirements pose some challenges on the system architecture addressed in the following sections.

For privacy purposes, numerous ATs, which are not linked between each other, will be assigned to the same vehicle to guarantee anonymity. This requirement contrasts the scalability requirement which demands an agile way of managing all certificates of the whole system.

Acknowledging that the anonymity requirement is abiding, and the pseudonym certificate architecture is hardly alterable, we seek to optimize the scalability requirement leaving untouched the anonymity practices. Scalability is crucial in the process of revocation of a misbehaving vehicle, as the communication of the identified misbehaving vehicle to the other vehicles in the system must be as swift as possible. The transmission of this information is performed with the broadcast of a CRL which grows in size proportionally to the number of revoked vehicles. Unfortunately, this can lead to huge CRLs which in turn impacts the bandwidth usage and processing overhead, and consequently having a negative impact on the scalability of the system and endangering its operation. For these reasons we analysed in the following chapter different ways of optimizing the CRL distribution.

4.1 Optimized CRL Distribution

The prompt revocation of misbehaving or malfunctioning vehicles is a crucial point in the framework of connected vehicles. Without revocation, malicious vehicles could be able to exploit the infrastructure and redirect traffic, generate false warnings and, more generally, disrupt the advantages of vehicular infrastructure. Anonymity and scalability are the requirements addressed in this section.

To guarantee the anonymity of vehicles, short-term certificates must not be linkable between one another. The total number of short-term certificates assigned to a vehicle depends on different factors, such as the validity period of the certificates and the number of simultaneously valid certificates assigned to a vehicle to protect its privacy. These factors are analysed in more detail in Section 4.4, however, the total number of certificates stored on a vehicle can grow very large. The Butterfly key expansion technique grants the ability of generating batches of unlikable certificates starting from a single request, diminishing the load on the requesting vehicle. The impossibility of linking of certificates between each other guarantees anonymity but on the other hand implies that certificates must be revoked one by one. When a malicious vehicle is detected, to prevent it from communicating with its surroundings and potentially take advantage of the infrastructure, all its active short-term certificates must be revoked.

The CRL, which is generated by the Revocation Authority and broadcasted to vehicles and RSUs, contains one entry for each certificate that is revoked. Vehicles and RSUs will use the information included in the CRL to drop all messages received that have been encrypted and/or signed with a certificate included in the CRL, without attempting to process them. Multiple certificates from a revoked vehicle can be identified as invalid, including both short and long-term certificates, and therefore included in the CRL.

Considering the number of short-term certificates assigned to a vehicle and the growing amount of revoked vehicles over time, the number of certificates to be included in the CRL can rapidly become hard to manage unless some optimization practices are put in place to ensure its timely distribution.

Three main approaches have been considered in the following subsections to address the creation, distribution and compression of the CRL.

4.1.1 Binary Hash Trees

V. Kumar et al. [36] proposed a different approach to standard revocation lists, replacing them with Certificate Access Lists (CAL). Vehicles are provided with encrypted certificates, specifically each batch of certificates (certificates to be used at the same time) is encrypted with the same symmetric key.

An entity called Certificate Access Manager (CAM) creates the encryption keys for each vehicle and each time frame. To do so, the CAM for each time frame creates a binary hash tree starting from a random seed. The depth of the tree is such that each leaf represents a vehicle. The value of each leaf, each identifying a vehicle, is the symmetric key for the decryption of the batch of certificates of the corresponding vehicle.

Each valid vehicle, starting from a node of the binary hash tree, is able to compute the children nodes and reach the leaf node having as value the decryption key for the active certificates. For this reason, it is not necessary to broadcast the whole tree but only the nodes necessary to calculate the leaf nodes of the valid vehicles.

Once the CAM has generated the hash tree for all the valid vehicles, it forwards the necessary nodes to obtain the decryption keys through the CAL. As a consequence, in the case where no vehicle is revoked, only the root of the binary hash tree needs to be forwarded. As the number of revoked vehicles grows, the average number of broadcasted messages is defined as: where is the number of revoked vehicles.

This approach brings the following advantages:

- It eliminates the need for bidirectional communication between the vehicles and the PKI.
- The revocation of vehicles is enforced on the PKI side and, as a consequence, once a vehicle is revoked it is simply prevented from accessing the stored certificates.
- A single decryption key from the binary hash tree decrypts the batch of certificates.

However, this approach has some drawbacks, especially regarding the integration in the structure described by ETSI [2]. The adoption of new entities and the transition from Certificate Revocation Lists to Certificate Access Lists poses an additional challenge.

4.1.2 Binary Hash Trees + Activation Codes

M.A. Simplicio et al. [37] extended and improved BCAM [36], starting from the same main concepts. The main difference lies within the generation of pseudonym certificates. In [36] pseudonym certificates are generated and then encrypted using a binary hash tree. In the same manner, in [37] the authors propose the generation of codes for each vehicle and each time period using a binary hash tree. This code is called “activation code”.

The activation codes are integrated directly in the *butterfly key expansion* process used to create the pseudonym certificates. Similarly to [36], the nodes of the binary hash tree which allow the calculation of the activation codes are broadcasted to the vehicles using RSUs. This new approach including activation codes brings higher performance and better security with respect to one based only on binary hash trees [36] at the cost of additional complexity on the PKI infrastructure. An increase of performance is achieved removing one layer of encryption and integrating the activation codes in the pseudonym certificates, saving numerous operation cycles.

The approach presented by M.A. Simplicio et al. [37] addresses one drawback of [36] design: it creates an extra point of collusion in the architecture. The CAM, like the Revocation Authority, learns which batch of encrypted certificates belong to the same vehicle and consequently, the CAM can collude with the Pseudonym Certificate Authority to violate those certificates’ unlinkability and, hence, the users’ privacy.

4.1.3 Bloom Filters

Bloom filters are a commonly used probabilistic data structure. They allow to efficiently compress information using the output of multiple hash functions, representing a set in a space-efficient way. The application of Bloom filters in vehicular networks has been presented in numerous articles [38][40][41]. Additionally, this data structure usage has also been studied in different contexts, such as smart grids [39] and as a OCSP replacement.

Because of the intrinsic probabilistic nature of Bloom filters, assessing the belonging of an item to a set could yield false positives but never false negatives. In the case of CRL distribution for VANETs, the set represented with the Bloom filter is the list of revoked certificates.

Therefore, a query on the belonging of a certificate to the set will:

- Never return that a revoked certificate is valid (false negative)
- Occasionally detect a valid certificate as revoked (false positive)

The latter occurrence, the false positive, has a defined probability can be tuned and it depends on the following variables: the size of the bitmap representing the filter (m), the number of elements in the set (n) and the number of hash functions used (k):

$$P_{false\ positive} = [1 - (1 - 1/m)^{kn}]^k$$

Bloom filters can be used as a drop-in replacement of standard CRLs, producing compression gains by, approximately, a factor of ten, depending on many factors. False positives, however, pose a challenge which can be addressed in two ways: backup certificates [38] and chained bloom filters [39][41].

4.1.3.1 Backup Certificates

Backup certificates can be used on the sender side to check whether the certificate the vehicle is about to use to sign a message will be detected as false positive. In the case of this occurrence, the sending vehicle will discard it and proceed to use a backup one.

As the possibility of false positives is quite low, only few backup certificates, to be added to the standard certificates are needed; approximately a couple of days' worth per year. In this manner false positives are handled before sending a message, moving the false positive challenge, along with the small overhead, on the sender side.

At the same time, malicious users are not able to take advantage of backup certificates because, once revoked, all the stored certificates, including the backup ones, are flagged as revoked with no possibility of false negatives.

4.1.3.2 Chained Bloom filters

The false positive challenge can be also addressed on the receiver side using chained bloom filters.

Chaining two bloom filters consists in sending one filter representing the revoked set of certificates and one filter representing the valid set of certificates. In this manner, the receiver is able to detect a false positive when the query returns that the certificate used to sign the received message is a member of both sets. In this case, the receiver will need to contact the PKI through the RSUs to assess the validity of the received certificate. However, the performance gain from the compression is halved due to the need of sending two bloom filters instead of one.

Moreover, the need to contact the PKI in the false positives case makes this approach less beneficial than using backup certificates.

4.1.4 Proposed approach

While the considered binary hash tree methods presented before proposing an interesting approach to address the CRL scalability issue, we propose to use Bloom filters to optimize the distribution of the CRL, for different reasons:

1. Bloom filters are extensively studied and have been integrated in the certificate revocation process in many fields, from smart grids and the Internet Protocol to the vehicular network itself.
2. Bloom filters allow for an almost seamless integration in the current infrastructure, without the need to add additional entities to the PKI infrastructure.
3. The significant compression gain derived from the space efficient structure justifies the introduction of a probabilistic data structure, moreover its drawbacks have been thoroughly studied and can be addressed (see Section 4.2 for more details).

To tackle the false positives probability, we propose to use backup certificates for different reasons:

1. The additional storage for the backup certificate is minimal, as the number of backup certificates, depending on the lifetime parameters of the certificate, can be in the order of a couple of days' worth each year.
2. The certificate check on the sender side does not put an additional overhead on the receiving side in case of false positives and does not require the receiver side to establish a connection to the PKI through the RSUs.
3. The performance of the filter is not substantially decreased by the introduction of more data in the CRL, as the usage of chained Bloom filters implies.

4.2 Optimized CRL Validation

In this validation, the benefits of Bloom Filter CRL compression have been analysed and, in particular, we investigated two main objectives: the optimal configuration parameters of the Bloom Filter and the compression gain of a Bloom Filter CRL with respect to a standard CRL.

Bloom Filters (BF) are a space efficient probabilistic data structure which allow to efficiently compress a set introducing, however, false positives. In the studied simulation the BF represents the list of revoked AT and it will return a false positive with a defined probability when interrogated on whether a specific AT belongs to the set. In the case of a false positive the BF will indicate that an AT is in the set and flagged as revoked even if it is not. The challenge of false positives is tackled using backup certificates on the sender side.

In this context, the BF contains a hashed value uniquely identifying an AT. The process of hashing consists of computing the SHA256 of the AT and then extracting the low order 8 bytes. This process yields the *HashedId8* of an AT which will be inserted in the BF.

The simulation has been performed on a laptop with the following characteristics:

- Intel Core i5-8265U CPU,
- 8GB of RAM,
- Windows 10 Enterprise 64 bit,
- Python version 3.7.8,
- The C library '*libbloom*' version 1.6,
- C extension module for Python.

4.2.1 Validation

The libbloom library allows us to create a BF defining two input parameters: maximum number of elements in the filter and expected false positive probability (FPP) for the full filter.

The first goal of this simulation was to analyse the BF features, the compression gain against the false positive probability, as a low compression gain would not justify the introduction of a BF and, finally, a high FPP that could challenge the effectiveness of the filter.

Figure 14 represents the output of the first simulation. Each point on the chart represents a simulation of N=1e6 items inserted in a BF created with different false probability input rate. The compression ratio grows with the false probability rate and because of this a good balance between the two parameters must be selected. We believe that a FPP higher than 0.04 would be impractical in the CARAMEL context and for this reason they have not been considered in the following analysis. While a compression of a factor of 8 would mean that the BF CRL grows by one byte at every AT revoked, with and expected FPP of 0.02, this scenario would require a high number of revoked certificates, mitigating the benefits of the BF. We expect lower FPP to be beneficial to the CARAMEL scope.

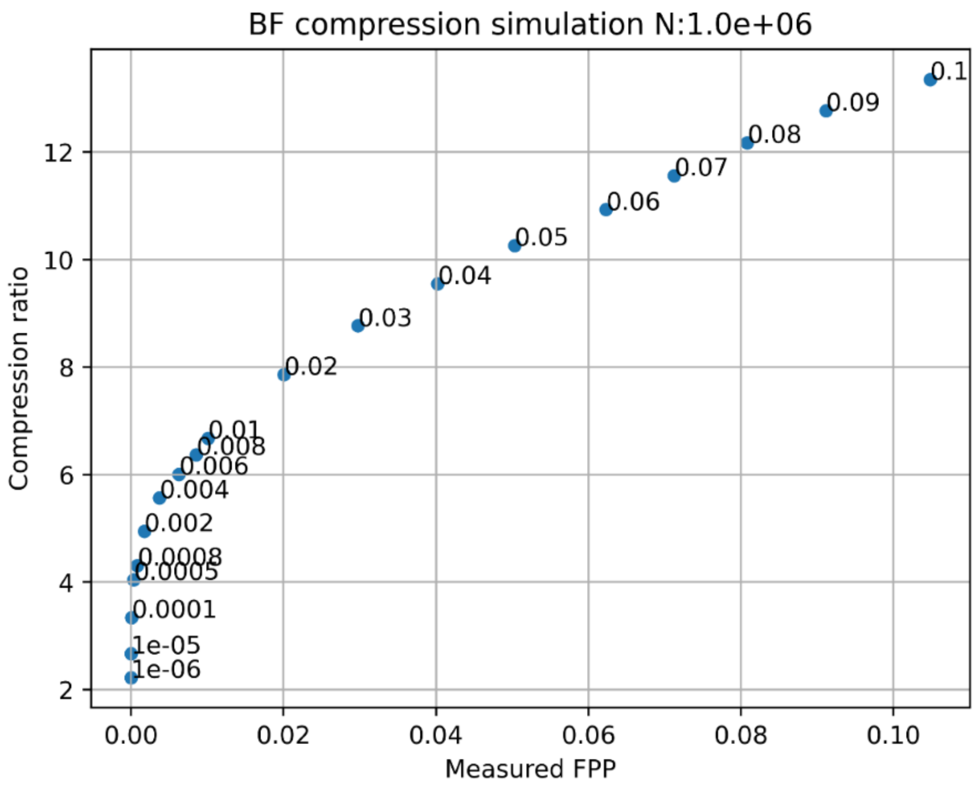

Figure 14: Compression gain against simulated (measured) false positive rate

Figure 15 shows the size comparison of different BF configurations against the standard CRL. The standard CRL grows linearly with the number of revoked vehicles, because one HashedId8 identifying an Authorization Ticket measures 8 bytes and Authorization Tickets are appended to the list as they are revoked.

In this simulation we used an adaptive technique to the Bloom Filter CRL approach, since the allocated space for a BF is fixed, we increase it step by step as the filter fills up. With this approach, the space of the BF is allocated when needed, minimizing the waste of bytes. When a new BF is created, the content of the previous one is transferred to the new one, in this way the new BF represents the whole CRL.

Once again in this graph we can observe the relationship between the FPP and the compression ratio (R). Conservative values of compression ratio (R less than 6.5) would guarantee a low FPP and consequently fewer backup certificates needed.

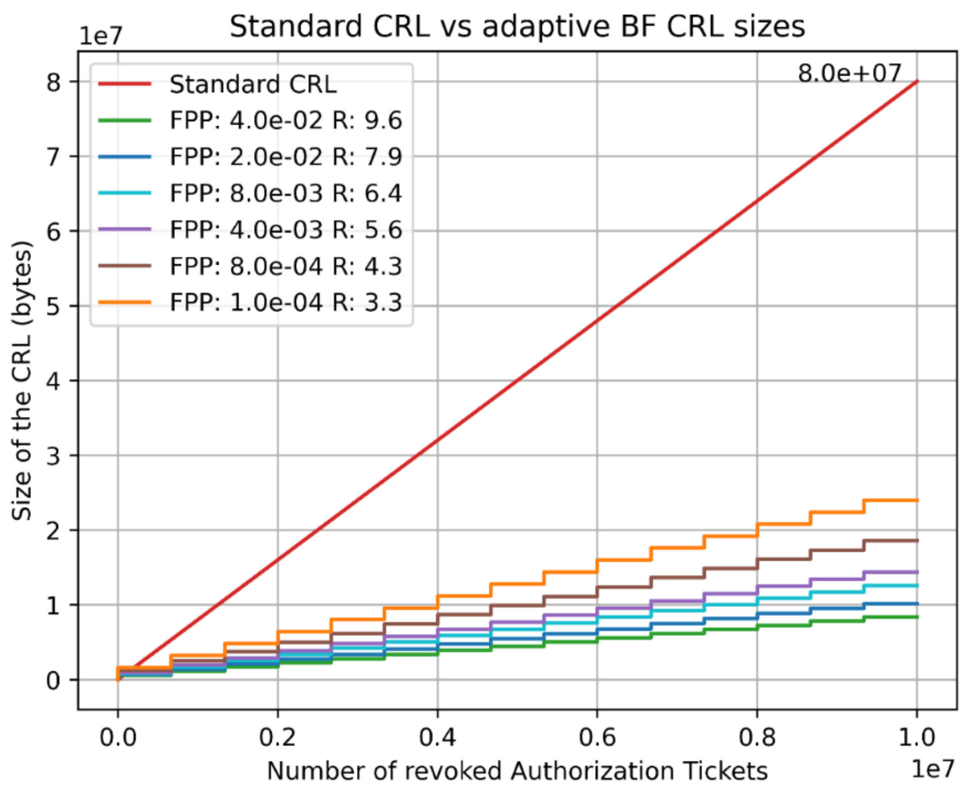

Figure 15: Standard CRL vs adaptive Bloom Filter sizes comparison for different input parameters of the BF

4.2.2 Backup certificates

To counter the Bloom Filter intrinsic characteristic of false positives, we propose to use backup certificates to replace the Authorization Tickets that will be discarded when a false positive is detected. A number of extra Authorization Tickets, defined by the FFP, is stored in a vehicle. The vehicle, before sending a message signed with an Authorization Ticket, checks the AT against the Bloom Filter CRL. If it triggers a false positive and the valid certificate is flagged as revoked, the vehicle discards the AT and uses a backup one.

The estimated number of backup certificates needed by the whole fleet of vehicles, considering a total amount of N=1e7 AT issued, is indicated in the following table, along with an adjusted compression ratio that considers the additional ATs.

Original compression ratio (R)	Compression ratio with backup certificates	False Positive Probability (FPP)	Total number of backup certificate needed for N=1e7
9.55	9.19	0.04	334983
7.86	7.71	0.02	203558
6.37	6.32	0.008	100494
5.57	5.55	0.004	57460
4.31	4.31	0.0008	14842
3.34	3.34	0.0001	2396

Table 1: FPP vs number of backup certificates

Similarly, Figure 16 represents the adaptive Bloom Filter CRL with the additional backup ATs. We can observe that the BF configurations with high compression ratio, and consequently high FPP, are the ones most affected by the increased number of backup certificates. Once again, we can see that a compression gain (R) lower than 6.5 would guarantee a good trade-off between the effective size of the CRL, the FPP and, consequently, the number of backup certificates needed.

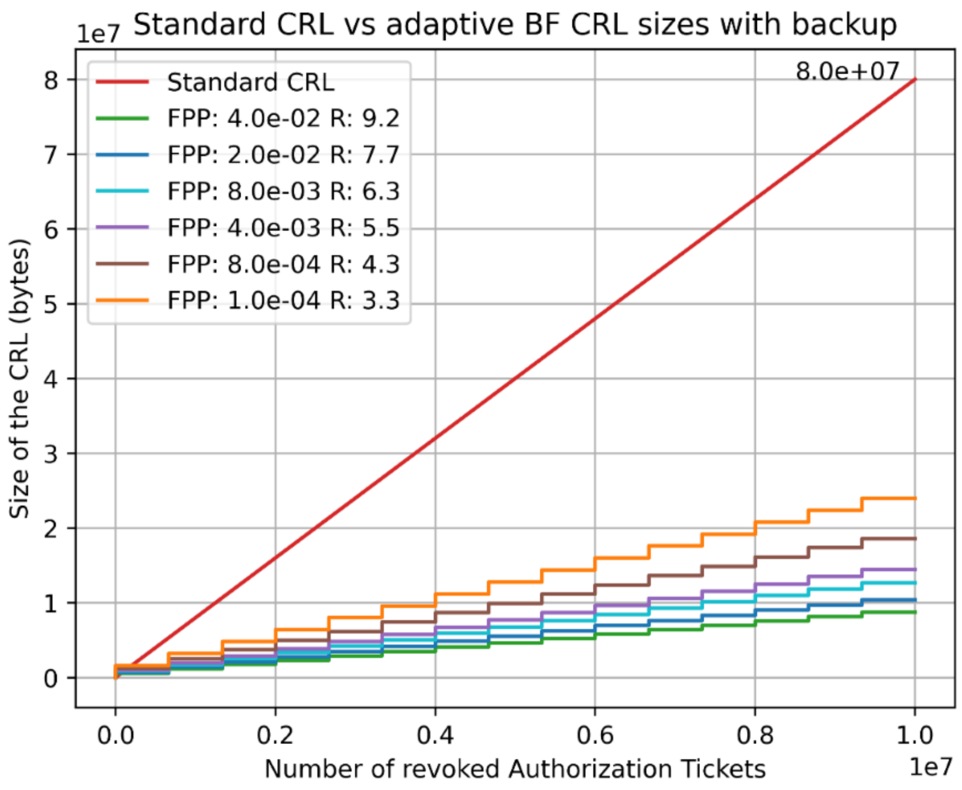

Figure 16: Adaptive Bloom Filter CRL accounting for the extra number of backup certificates

4.3 Federated Trust Management

We investigated a federated trust management system architecture to address the scalability requirement. Partitioning the whole PKI infrastructure into federations, each mapping a defined geographical region, allows to keep under control the number of certificates in each region.

Each federation manages the vehicles present in the corresponding region, issuing ECs and ATs and broadcasting CRLs. The certificates used by the vehicles identify the regional authorities and are valid only in the geographical region of the federation.

When a vehicle travels from one federation to another, it requires a new set of ATs to transmit valid messages to its surroundings. To do so, it contacts a nearby RSU belonging to the new federation and it relays the request to the Authorization Authority (AA). Internally, the AA checks the validity of the certificate received from the vehicle and its revocation status contacting the federation of origin of the vehicle. If the checks are successful, the AA generates and forwards the new ATs for the new region to the vehicle.

4.4 Attacks on Authorization Tickets Tracking

Each V2X message sent by a connected vehicle must be signed with an Authorization Ticket (AT). These ATs are anonymous and do not reveal any information about the vehicle or the driver. The value that ATs brings is that a V2X spoofer could not recognize the driving (and living) patterns of a targeted person.

An advanced spoofer could try to track a vehicle that sends the V2X messages signed with the same AT. This spoofer would obtain the trajectory of the car and this information could be cross-correlated with additional data obtained by other means that would reveal the identity of the driver. For example, the starting point of the trajectory would be the driver's home address and the end point would be the work place. The Authorization Authority solves this problem by assigning periodically a set of ATs to each connected vehicle. In this way, the connected vehicle can change the AT at any time and break the tracking line that the spoofer has been following.

The spoofer could go further and try to reconstruct the trajectory of a vehicle that has signed the V2X messages with different ATs. This is not an impossible task, since the connected vehicle sends the V2X messages in periods ranging from 100 to 1000 milliseconds. That means that a vehicle running at 39 km/h will only advance a few meters during the time in between two messages sent (1 meter if the time is 100 milliseconds and 10 meter if the time is 1000 milliseconds). This proximity makes it easy to relate two V2X messages with different ATs to the same vehicle, especially when there are not more vehicles around them.

A most extreme case would be when the spoofer is a trained ML algorithm that considers all the information available in the V2X messages to decide that two V2X messages with different AT belong to the same vehicle. This is the problem that is tackled in the present section and that we will try to mitigate.

4.4.1 Attack Mitigation with the AT Scheduler

This section proposes an effective AT scheduler architecture that decides the best moment to change the AT of a target vehicle to avoid being tracked by an attacker. The decision is made by evaluating how easily it is to track the car using the information contained in its V2X messages. The scheduler is designed to be placed in the vehicle’s anti-hacking device, and it takes into account a buffer of old V2X messages from both the owner’s car and the surrounding vehicles at a given time and place.

More precisely, a tracker tries to associate each V2X message sent by the target car with a subgroup of old messages that were sent by the same vehicle. Although the length of this subgroup can be variable, in practice, the tracker only needs to link the new message with the latest message sent by the target car in order to track it. The system stores a buffer of old V2X messages coming from the surrounding vehicles and the target car itself. The system is trained to discern which of those old messages correspond to the same target vehicle. This way, it is possible to evaluate periodically how well the car can be tracked by an external spoofer. If the target messages are associated with the correct subgroup with a high score, it means that the car can be easily tracked.

The AT scheduler is composed of three main modules:

- A **V2X message candidates selector**, that selects a group of N old messages that are most likely to belong to the target vehicle.
- A **V2X message tracking scorer**, that associates the incoming message with a subgroup of the N candidates with a certain score.
- An **AT change decision engine**, that decides the best moment to change the AT of the V2X message according to a buffer of tracker scores, the number of remaining ATs and the time to get a new pack of ATs.

Each of the modules are further explained in the following subsections.

The AT scheduler architecture has been designed following a modular approach: a pipeline of two independent blocks that work simultaneously. The first block receives the tracking messages of the surrounding vehicles and from the car itself and uses a tracking algorithm to produce a score that asses the tracking difficulty, simulating a spoofer. These scores are used in the AT change block. This block decides when is the best time to change the AT given the scores in time, the remaining valid ATs in the car and the time to receive new ATs.

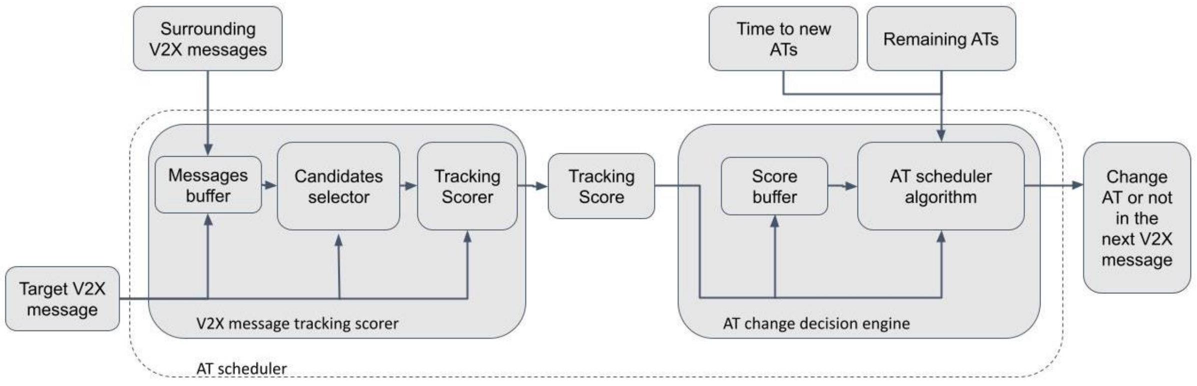

Figure 17: AT Scheduler Architecture

4.4.2 AT Scheduler Modules

This section provides an in-depth description of the three main modules that form the AT scheduler. All of the modules are used once for each incoming V2X message from the target car.

4.4.2.1 V2X Message Candidate Selector

The candidate selector module is responsible for providing a batch of passed messages (candidates) that have chances to belong to the same car as the incoming message (target), as well as some features that can be used by the tracker to link them. To select the candidates, the system stores a list of old V2X messages that were sent by the surrounding cars along with those that were sent by the target car. This list is periodically updated to keep only the last M messages, so older messages are discarded to maintain a manageable amount of data.

The candidate selector takes as input all the messages in the buffer. As all ATs should be unique, if the incoming AT matches one of the old messages it can be directly linked without further analysis. If not, the module computes some features of the candidates that can be useful to link them to the target. All the features are assumed to be computed using information that is available in the V2X messages. These features may include, but are not limited to:

- The time difference between the candidate and the target.
- The variation between the target's position coordinates and the expected position that the candidate should have at the time of the target.
- The variation between the target's velocity and acceleration components and the candidate's expected values of those components at the time of the target.

Finally, the candidates are sorted by the values of their features and the selector returns the N candidates with the smallest variations so that they can be analysed by the tracker.

4.4.2.2 V2X Message Tracking Scorer

Given a set of candidates and their features, the tracking scorer works as a regression algorithm that chooses to which candidate a target message is linked, meaning that the two messages were sent by the same vehicle.

The proposed method to implement the tracking scorer is a Random Forest [43], which is an ensemble learning method consisting of a fixed number of decision trees. Each decision tree evaluates a random subset of the available features with specific thresholds to decide whether the candidate messages belong to the same vehicle as the target message. For each candidate, each tree outputs a binary value $y \in \{0,1\}$ and the Random Forest outputs a single score $Y \in [0,1]$ representing the average prediction of the decision trees.

One of the main advantages of Random Forests is that they are less prone to overfitting than a single deep decision tree, thanks to the creation of random subsets of features. Ensemble methods can produce more accurate results than any of their individual predictions while reducing the variance of the outcomes. Random Forests are also less computationally demanding than other common AI classification systems, such as Neural Networks.

The Random Forest is trained using the CARAMEL V2X dataset described in D5.2, which is based on the vehicular mobility dataset [44]. The dataset contains V2X messages of multiple cars with random periods of 100 ± 50 ms, each one containing the following variables:

- Message timestamp.
- Car id.
- x and y coordinates of vehicle position.
- x and y components of vehicle velocity and acceleration.
- Module of the vehicle velocity and acceleration.

4.4.2.3 AT Change Decision Engine

The AT change decision algorithm approach is modelled as a well-known mathematical problem known as optimal stopping, also called the marriage problem, secretary problem or best-choice problem. Selecting the best time to change the AT is crucial to minimize the spoofing tracking capabilities. In the optimal stopping scenario, a decision-maker observes inputs that evolve in time and involve some randomness and decides when is the best moment to perform an action given the known inputs. A time limit to make the decision also needs to be set, otherwise the decision-maker would be observing the inputs indefinitely looking for the best moment to perform the action. In this system, the decision-maker is the AT change decision algorithm, the inputs are the scores given by the V2X tracker scorer, the action is to change the AT and the time limit is a function defined below.

It happens that the number of actions (changing the AT) that can be performed depends on the number of non-used AT that the vehicle still has. Once all the AT of the vehicle have been used, not more AT changes will be possible until the vehicle receives new AT from the Authentication Authority. To distribute the AT equally on time, the time limit to change the AT has been defined as a function of the remaining non-used AT in the vehicle and the time left to get new AT from the Authentication Authority:

$$Time\ limit\ = \frac{time\ to\ new\ ATs}{Remaining\ ATs}$$

It is important to highlight that the time limit will be updated after each time that the AT is changed. It has been proven [45] that the best decision can be made with a probability greater than 36%. The strategy to be followed is to spend the 37% of the available (time limit) observing the tracking scores and get the maximum value of these scores. Then check the following scores until getting a value that is greater than the maximum value obtained during the observation time. If any greater value appears during the rest of the available time and the time limit is reached, the system forces an AT change.

5 Architecture and Design of the CARAMEL PKI Toolbox

5.1 Overview

Figure 18 provides a high-level view of the V2X system architecture where the PKI Toolbox components are located. The PKI core is located in the Remote Infrastructure (green box), while the PKI Client runs inside the connected vehicle (red box).

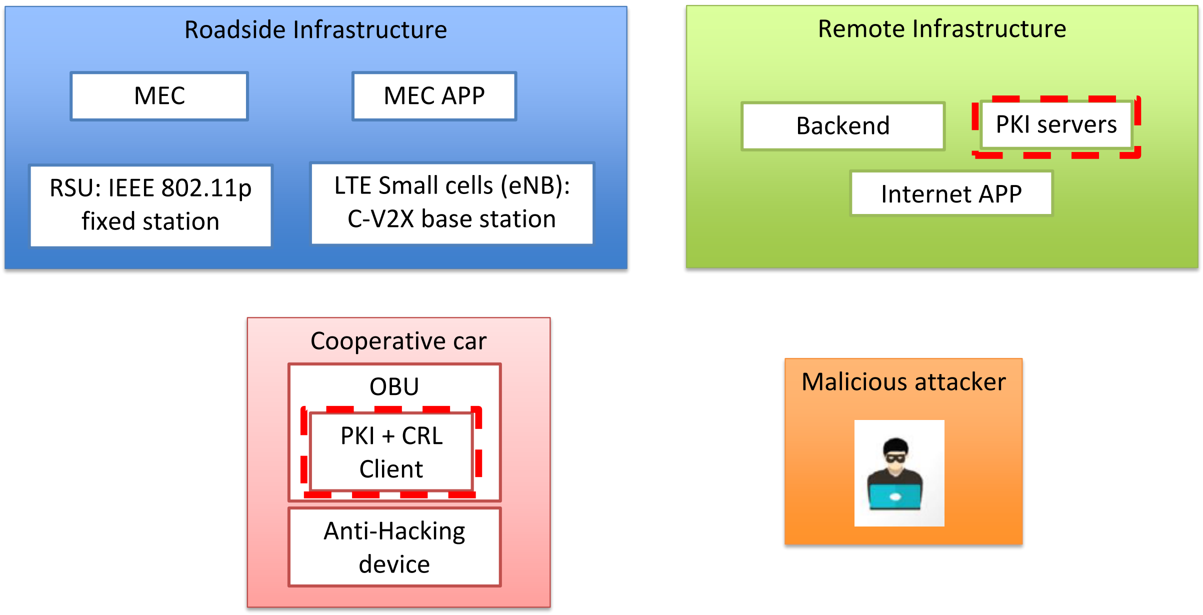

Figure 18: V2X Global Architecture

Extending the previous architecture, Figure 19 illustrates in detail the PKI Toolbox architecture, including its main components that are described in detail in the subsequent sections. The figure also shows the distinction between the Infrastructure services, which are supporting services, and the core services which are providing the business logic of the PKI Toolbox.

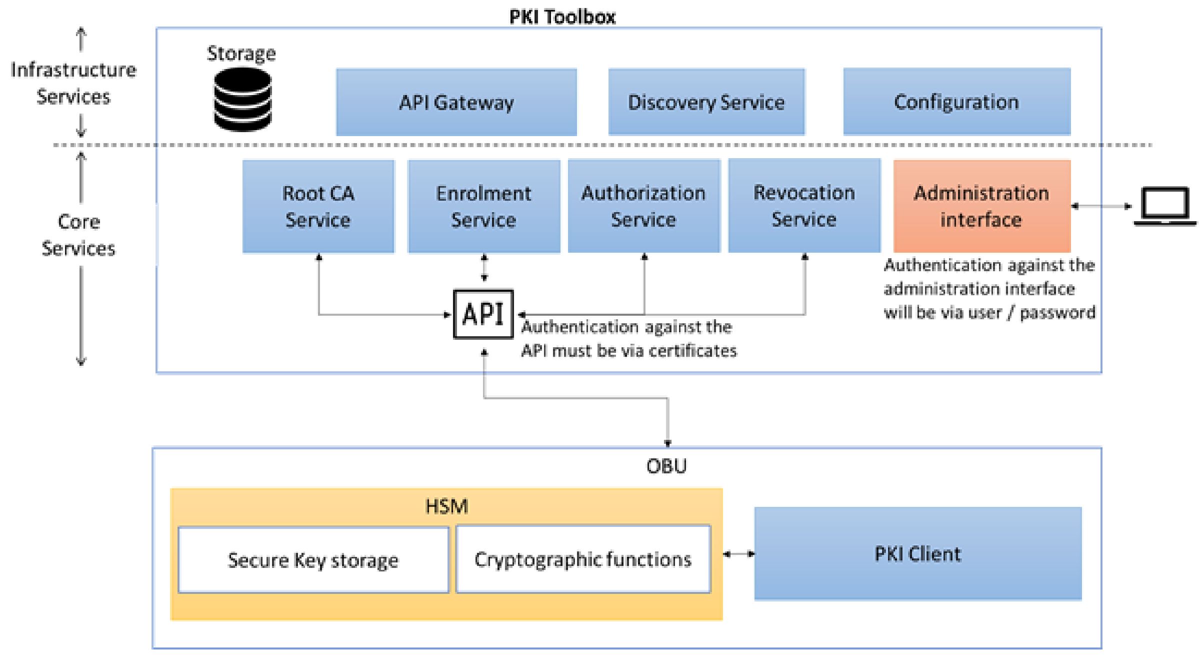

Figure 19: PKI Toolbox Architecture

5.2 Component Model

In the following section various components of the PKI are described. The PKI consists of several modules which handle different aspects of the PKI such as infrastructure services and core services.

5.2.1 Infrastructure Services

This set of services are in charge of providing basic supporting functionalities for managing the lifecycle of the core services. They do not implement the business logic of the PKI Toolbox but provide essential capabilities for its operation.

5.2.1.1 Configuration Service

The configuration service is a microservice aiming to provide centralized configuration storage for all the PKI microservices. Each microservice can have its own configuration storage that will include all configurable parameters such as configuration of the application, database configuration, configuration of the APIs to communicate with the other services, etc. This could be seen as an externalization of the management of the properties/resource file out of the microservice project to an external entity service, which allows changes to any property without needing the re-deployment of the service using it. Furthermore, the configuration server also allows the possibility of assigning distinct configuration for different environments, separating in this way the configuration of the development and production environments.

Before accessing its configuration, the microservice must provide its instance id which is composed by a combination of the microservice name and the environment in which the service is executed. At this moment, the configuration server retrieves the configuration and provides them to the requesting microservice. To protect the access to the configuration, the configuration server will be secured and include a role-based access control mechanism to guarantee that only the authorized microservices are able to access to their configuration.

In the context of the CARAMEL project, we have used Spring Cloud Config Server [46] for the implementation of the PKI Toolbox configuration service. Spring Cloud Config Server is one of the most popular implementations which enables the storage of the microservices configuration for multiple environments in a Git or a SVN Repository, including interesting capabilities such as the auto-refresh of the configuration.

5.2.1.2 Discovery Service

Microservices allows the scalability of the PKI Toolbox architecture by enabling the auto-scaling of a number of microservice instances depending on the system load or configuration requirements. This auto-scaling capability brings a lot of benefits and flexibility to the system architecture but also opens different issues to be addressed such as knowing how many instances of a microservices are running at a given moment and how to distribute the load among them. Here is when the Discovery Service enters into the scene to address those issues. Basically, the Discovery Service offers a kind of naming service where all the running instances are registered. Each microservice only needs to know about the location of the discovery service and will use it to get the location of other microservices. Before calling another microservice, a service must ask the Discovery Service about the available instances of that concrete microservice for obtaining the references.

In the context of the CARAMEL project, we are using Netflix Eureka server [47] for the implementation of the service registry server and Eureka clients (included in all microservices) which will register themselves and discover other microservices. Netflix Eureka is a popular specialized open-source option for implementing this type of service, but we also evaluated other options such as Consul [48]. We finally selected Eureka because it offers excellent integration with Spring Cloud, is implanted on the JVM and can be bootstrapped automatically in one of Spring Cloud’s auto-configurations. Consul otherwise is implemented in Go.

5.2.1.3 API Gateway

In a layered architecture, such as the one we are following for the PKI Toolbox, there are specific functionalities that are required in all layers, such as logging, security (authentication and authorization), performance, auditing and rate limiting. Also, the intrinsic granularity of microservices implies that they typically offer fine-grained APIs, which means that clients have to interact with multiple services to complete an action. Furthermore, the number of active instances, their locations, and the services partitioning may change over the time. To decouple clients from this complexity, it makes sense to implement all those cross-layer features in a common way. All these cross-cutting concerns can be jointly addressed in an effective way by means of the implementation of an API Gateway that will provide a single-entry point for all clients to access to those cross-layer functionalities. The API Gateway will handle all requests – both internal and external - either, forwarding them to the appropriate microservices or by fanning out to multiple microservices.

For the implementation of the API Gateway inside the PKI Toolbox, we have selected Spring Cloud Gateway [49] which is built on top of the Spring Ecosystem. As an alternative, we evaluated the option of using Zuul [50] but finally we decided on Spring Cloud Gateway because it offers better performance in the latest versions of Spring Cloud, it is better implemented in this ecosystem and allows long-term connections.

5.2.2 Core Services

In this section we describe the PKI authorities which provide the core services and their functions.

5.2.2.1 Root CA Service

This service is in charge of implementing the Root Certification Authority (RCA). The RCA is the root of trust for all certificates within the PKI hierarchy. Additionally, the RCA is also responsible for issuing certificates to the sub-CAs (Enrolment, Authorization and Revocation Authorities) which are used to sign derived certificates (enrolment and pseudonym certificates) generated by the sub-CAs. Those certificates “inherit” the trustworthiness of the root certificate.

Both, the root CA Certificates and the certificates of the subordinate certification authorities shall be of type *EtsiTs103097Certificate* as defined in ETSI TS 103 097 V1.3.1 [31]. The generation of those certificates is a manual process done by PKI administrator during the setup of the infrastructure. The following use case diagram provides a high-level overview of the functions provided by this service.

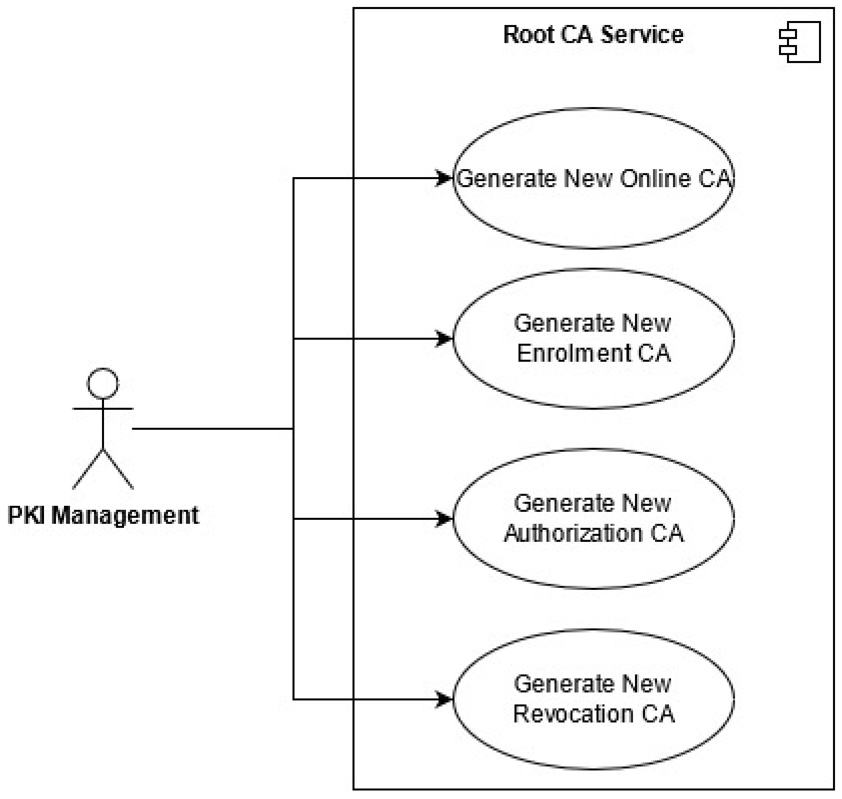
Root CA Service
Generate New Online CA
Generate New Enrolment CA
PKI Management
Generate New Authorization CA
Generate New Revocation CA

Figure 20: Root CA Service UML diagram

5.2.2.2 Enrolment Service

The Enrolment Service is in charge of implementing the Enrolment Authority (EA) which in turn is responsible for authenticating an ITS-S and grant it access to the ITS communications. This grant is assigned during the Enrolment phase in which a dialogue between the ITS-S and the Enrolment Authority is established to authenticate the car’s OBU canonical ID or certificate (Bootstrap Certificate). Once the ITS-S identity has been authenticated, the EA will make use of the canonical credentials to derive the enrolment credentials (Enrolment Certificate). In the same way, the enrolment credentials can be renewed or reissued once expired. In this case, the EA will make use of the previously assigned enrolment credential to generate the new ones. Additionally, the EA will implement also the so-called Revocation of Trust, for which a misbehaving ITS-S can be removed from the system by revoking its enrolment credentials and consequent short-term identities. This process will be done in coordination with the Revocation Authority to update the corresponding CRL (see Section 5.2.2.4).

The following use case diagram provides a high-level overview of the functions provided by this service. The Enrolment management function included in this diagram offers the possibility of manually managing the ITS-Ss, including its registration, status and permissions.

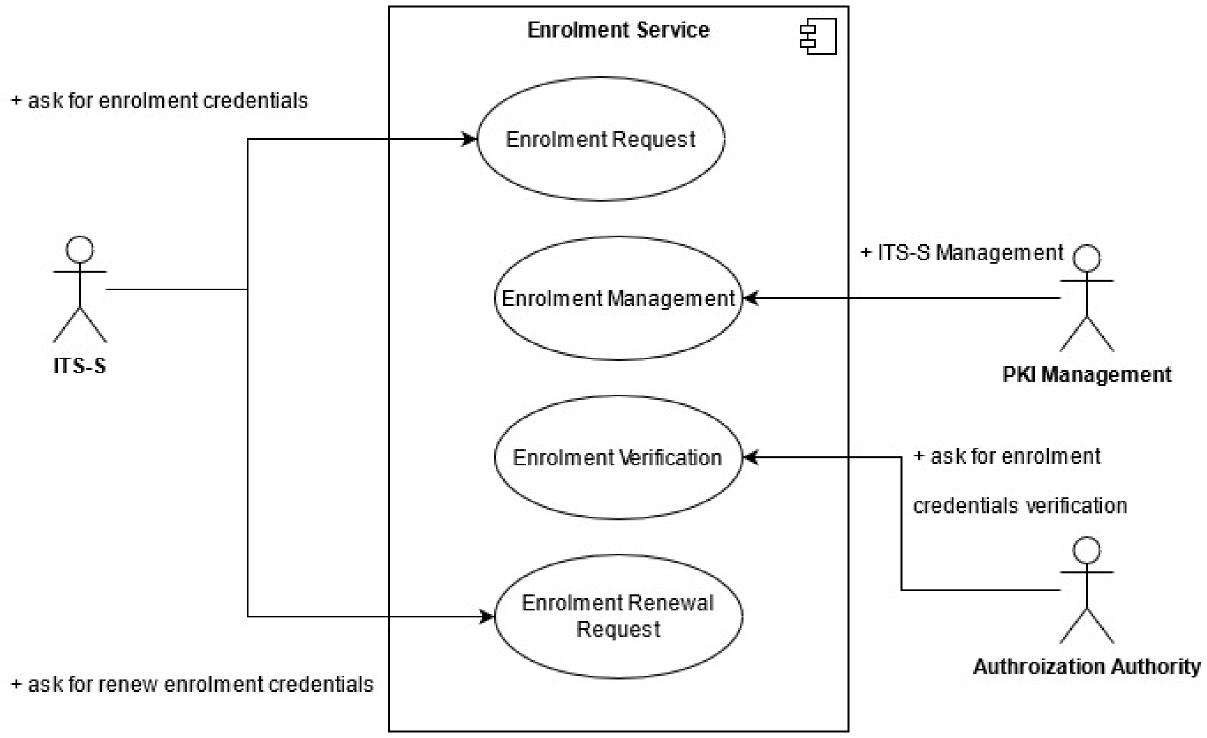
Enrolment Service
+ ask for enrolment credentials
Enrolment Request
ITS-S
Enrolment Management
+ ITS-S Management
PKI Management
Enrolment Verification
+ ask for enrolment credentials verification
Enrolment Renewal Request
Authroization Authority
+ ask for renew enrolment credentials

Figure 21: Enrolment Service UML diagram

5.2.2.3 Authorization Service

The Authorization Authority (AA), implemented by the Authorization Services, provides to the ITS-S with an authoritative proof which states it may use specific ITS services. This authoritative proof is the so-called Authorization Ticket (AT), also known as short-term certificate or pseudonym certificate, which guarantees the anonymity of the ITS-S identity and is standardized in Security Header and Certificate Formats by ETSI TS 103097 [31]. Each AT specifies a particular authorization context which comprises a set of permissions, such as the permission for broadcasting messages from a particular message set.

Additionally, thanks to the AT, the identity of the ITS-s can be referenced without revealing the identity of the vehicle or its driver.

The authorization phase starts once the ITS-S obtained its enrolment certificate. With it, the ITS-S generates the candidate AT key pair and the AA answers with certificates derived from the public key provided by the ITS station. The certificates will be stored and used afterwards for packet encapsulation while the private will reside within the HSM. The ATs and private keys stored within the HSM have to be available by the network layer (V2XCom-11pRSUs and V2XCom-CV2XRSU) to proceed with the message encapsulation. For instance, CAM or DENM messages are signed using the AT and placed in a data structure before being encoded using ASN.1/COER and processed by the network layer.

The following use case diagram provides a high-level overview of the functions provided by this service.

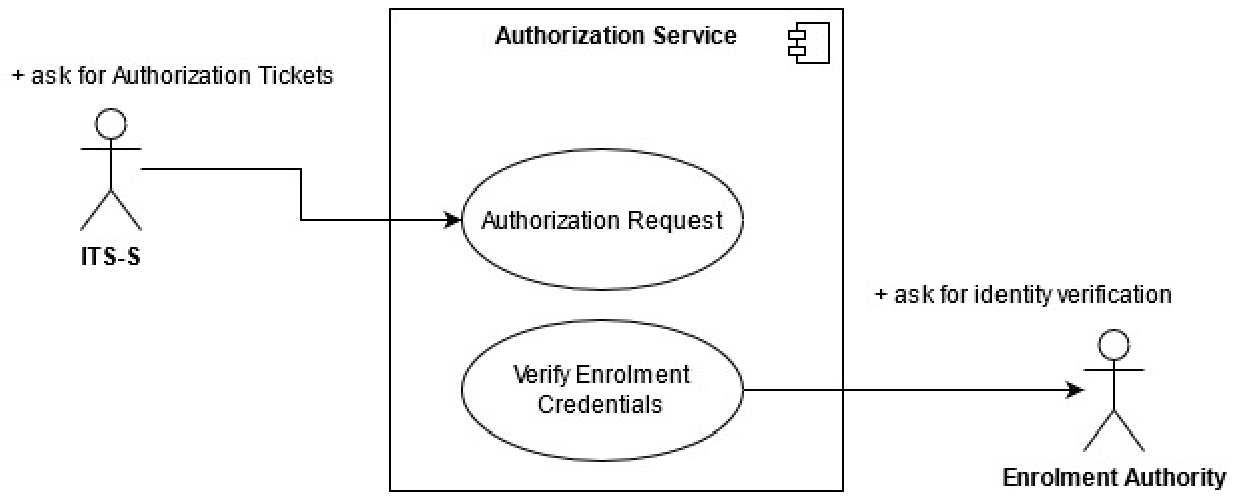
Authorization Service
+ ask for Authorization Tickets
ITS-S
Authorization Request
Verify Enrolment Credentials
+ ask for identity verification
Enrolment Authority

Figure 22: Authorization Service UML diagram

5.2.2.4 Revocation Service

The revocation service addresses the task of managing the detected misbehaving vehicles and it is performed by the Revocation Authority (RA). The RA is responsible of generating the Certificate Revocation List (CRL) identifying the revoked vehicles. The MEC Revocation Service detects and communicates to the RA the misbehaving vehicles and the RA proceeds to update the CRL contents with the active Authorization Tickets (AT) corresponding to the detected misbehaving vehicles.

The CRL size can grow large in size as the number of revoked vehicles increases, for this reason the CRL is constantly updated to contain only the ATs corresponding to the current time frame and in the RA's domain. Additionally, to reduce the size of the transmitted CRL, its contents are compressed using a Bloom filter (see Section Bloom Filters). Bloom filters are a space-efficient probabilistic data structure that allow vehicles to efficiently assess the belonging of a certificate to the set identifying the revoked certificates.

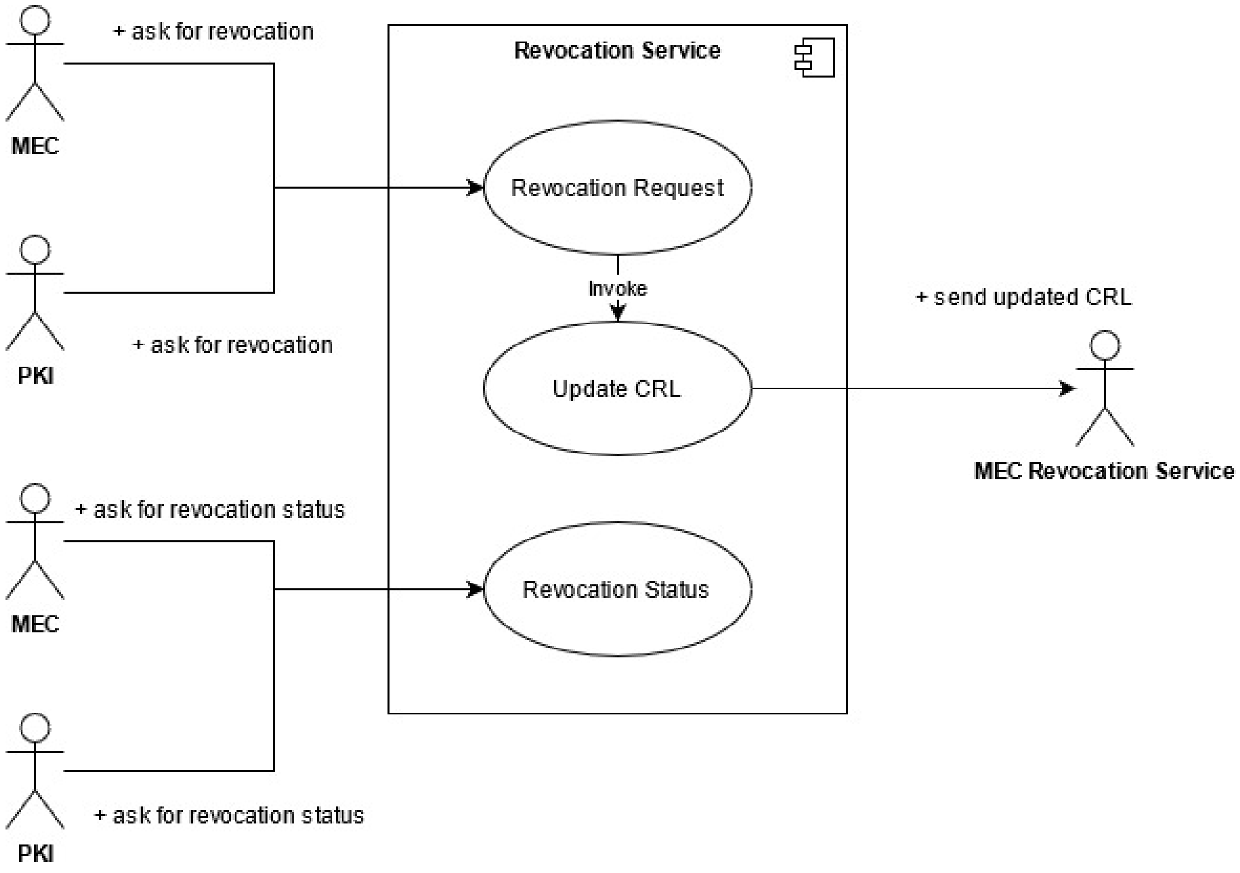

Figure 23: Revocation Service UML diagram

5.2.3 MEC Revocation Service

The purpose of this MEC Revocation service is to facilitate the communication flow of the PKI core system and its clients. It receives revocation requests and provides Certificate Revocation List (CRL) dissemination and CRL synchronization capabilities to the MEC so that not all PKI transactions go to the PKI core itself.

It communicates with the Registration Authority (RA) by forwarding Authorization Tickets (AT) revocation requests from different sources (e.g. OBU's or other MEC services), by receiving and distributing the CRL that the RA will update with these new requests to all the interested clients.

Besides the focus on CRL dissemination, the service also provides a synchronization mechanism for OBU's and other clients to request the current CRL. This CRL synchronization mechanism comprises two steps: the first where it broadcasts the current version of the CRL it contains, so that all clients can know the current version being propagated. In case any client has a previous or different version, it can make an (REST-API) request to ask for the current version being stored in the MEC.

This service does not contain any capability to decide on the actual revocation request or to provide an idea of any linked AT to the one being requested since this can only be done by the RA itself. It has two forms of communicating: MQTT and REST-API, both are used in the two tasks.

5.2.3.1 Components

The image below provides an overview of the MEC Revocation service. It contains two different communication channels: APIs and MQTT. All the use cases for each interaction between the MEC PKI and other entities are described in the use case section of this document.

APIs are used for direct communication with the MEC PKI. For example, MEC PKI clients to request for CRL updates to MEC PKI or the synchronization with the PKI Core itself. The MQTT channels ought to be used for dissemination purposes when MEC-PKI needs to communicate certain messages to all its clients.

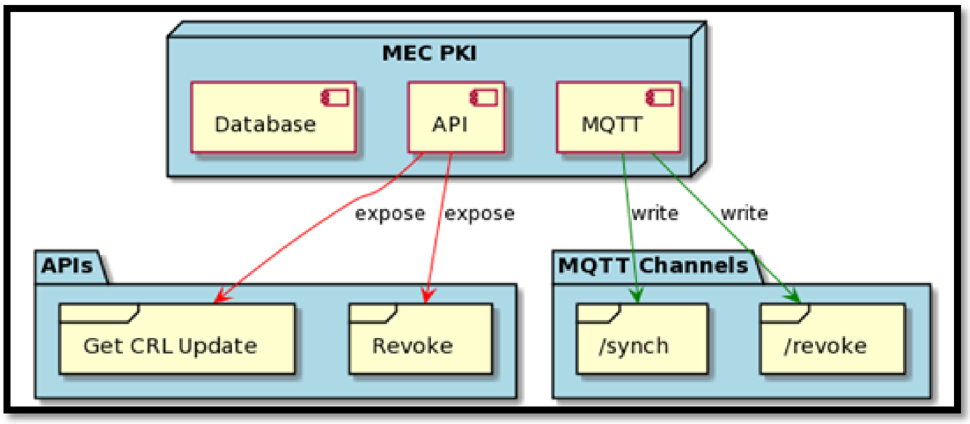

Figure 24: MEC Revocation Service Overview

5.2.3.2 Revocation

In the figure below is represented an AT revocation flow between the MEC revocation service, the PKI core and an OBU. The process starts when the MEC receives an AT revocation request from an OBU's (can also come from other sources such as other services in the MEC) and passes those requests to the PKI Core (main infrastructure). Note that it does not make any assumption whether this request is valid or not, it only serves as a forwarding point for these requests, thus no changes are made to the current CRL.

The RA then receives these requests and updates the CRL contents with the AT corresponding to the detected misbehaving vehicles and then compiles a new CRL to be distributed to the MEC. The MEC updates its own CRL and proceeds to deliver it to a specific MQTT channel which all interested parties are listening only to receive CRL updates. These interested parties receive the CRL and update their own CRL with the new info which may contain the revoked AT or more, since there is a possibility that with that revocation others approved ATs might lose their approved status.

All communication between entities has a prefix of MQTT or API depending on the channel of communication it depends on. The requests with no prefix can be viewed as internal processes the systems are intended to perform to complete the scenarios described.

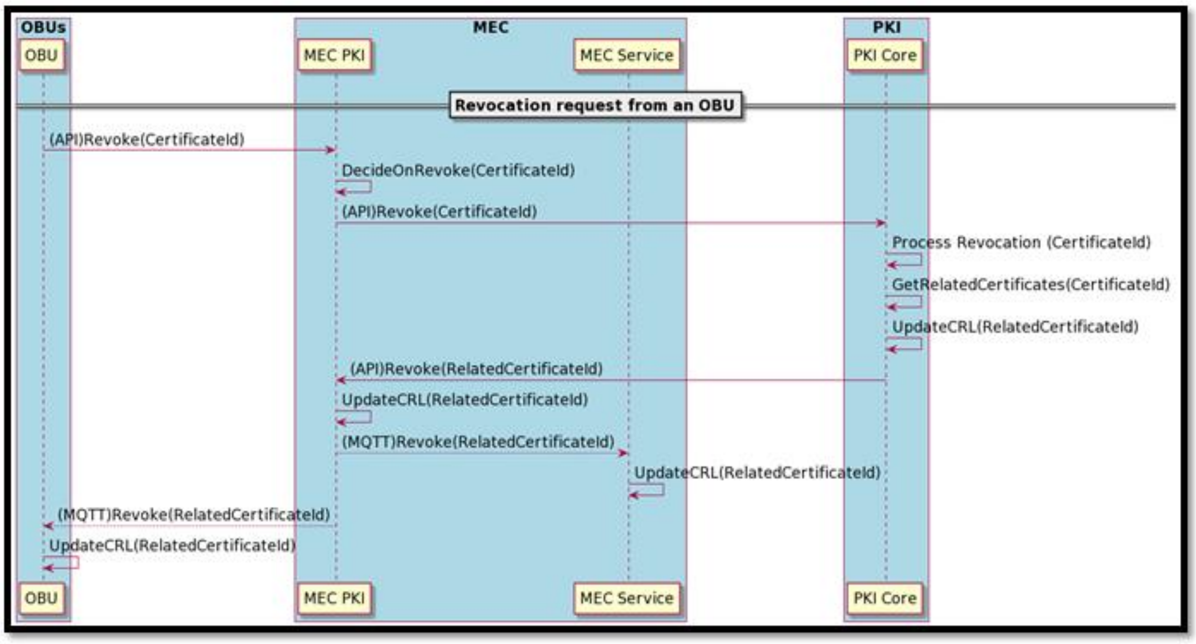
OBUs
OBU
MEC
MEC PKI
MEC Service
PKI
PKI Core
Revocation request from an OBU
(API)Revoke(CertificateId)
DecideOnRevoke(CertificateId)
(API)Revoke(CertificateId)
Process Revocation (CertificateId)
GetRelatedCertificates(CertificateId)
UpdateCRL(RelatedCertificateId)
(API)Revoke(RelatedCertificateId)
UpdateCRL(RelatedCertificateId)
(MQTT)Revoke(RelatedCertificateId)
UpdateCRL(RelatedCertificateId)
(MQTT)Revoke(RelatedCertificateId)
UpdateCRL(RelatedCertificateId)

Figure 25: MEC Revocation Service Sequence Diagram

5.2.3.3 Synchronisation

The Figure 26 demonstrates the actual flow of the synchronization process. The first step is a periodic update of the MEC-PKI broadcasts into an MQTT channel, since there is already an MQTT broker in the MEC, this way of communication is the most easy and reliant to guarantee all interested parties can receive the periodic notice with minimal effort for this service.

This update will only contain the current version of the CRL compiled by the PKI core. After receiving this version update, a PKI client requests the CRL to the MEC-PKI using a REST-API and updates its CRL with the MEC current version.

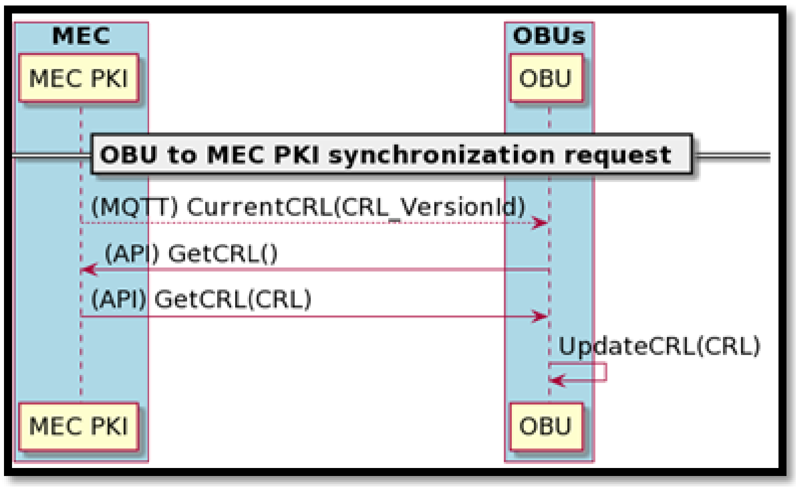
MEC
MEC PKI
OBUs
OBU
OBU to MEC PKI synchronization request
(MQTT) CurrentCRL(CRL_VersionId)
(API) GetCRL()
(API) GetCRL(CRL)
UpdateCRL(CRL)

Figure 26: Synchronisation Sequence Diagram

5.2.3.4 PKI Client

The PKI Client is the component acting as a bridge between the PKI core services and the V2X communication services running on the vehicle, with the aim of making them secure through the use of the distributed certificates. This component will run on the On-Board Unit (OBU) described in D3.6 and interact directly with the Hardware Security Module (HSM) which will be providing secure storage for the cryptographic keys and cryptographic operations.

When started, the PKI Client will be in charge on initiating the Enrolment phase by using the Canonical ID, profile and credentials pre-stored in the HSM. As a result of the vehicle enrolment, and assuming the process is completed successfully, the PKI Client will receive the Enrolment Credentials (Enrolment Certificate) which will be also stored in the HSM.

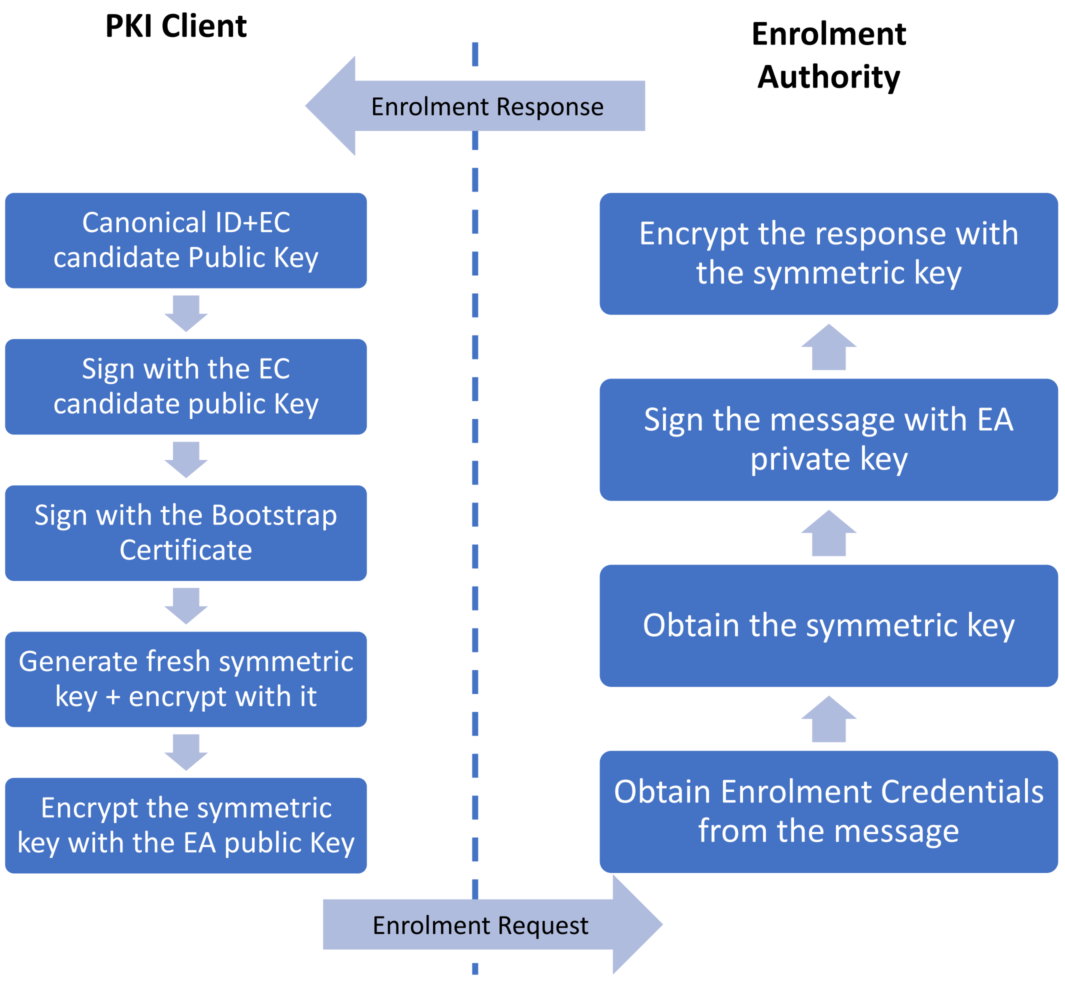
PKI Client
Enrolment Authority
Enrolment Response
Canonical ID+EC candidate Public Key
Sign with the EC candidate public Key
Sign with the Bootstrap Certificate
Generate fresh symmetric key + encrypt with it
Encrypt the symmetric key with the EA public Key
Enrolment Request
Obtain Enrolment Credentials from the message
Obtain the symmetric key
Sign the message with EA private key
Encrypt the response with the symmetric key

Figure 27: Enrolment process

To send the Enrolment Request, the ITS-S will perform the following activities (see Figure 27):

1. It provides its identity (canonical ID) and the Enrolment Certificate (EC) candidate public key (newly generated keypair for the new EC).
2. It signs this data with the candidate private key (use the private key from the new keypair) to prove it possesses the key pair for which enrolment is asked.

3. It signs with its current EC (or registration key = technical key) (use the current Bootstrap certificate or old EC in case of rekey[1] procedure) to prove its claimed identity.
4. It generates a fresh symmetric key which is used to encrypt.
5. To protect the encryption key (the one generated in the previous step), it will be encrypted with the Enrolment Authority public key and annexed to the message. In this way, only Enrolment Authority can retrieve the symmetric key and decrypt the message.

As a result of the Enrolment Request, the Enrolment Authority will generate the Enrolment Response following these steps:

1. Once the message is validated, the Enrolment Authority retrieves from the message the Enrolment Credentials.
2. The authority signs the message with its private key to prove its claimed identity.
3. The authority encrypts the response with the symmetric key used for the request (the one generated by the requesting vehicle and encrypted using the Enrolment CA certificate).

Next, the PKI Client will start the Authorization phase, using the Enrolment Credentials previously acquired. Once finished, the PKI Client will be provided with a set of Authorization Tickets that will be used as short-term credentials for establishing secure and anonymous V2X communications. The Authorization Tickets will be also stored in the HSM.

Additionally, the PKI Client will be also in charge of communicating with the MEC Revocation Service (see Section 5.2.3) to receive the updated CRL through the MQTT channel.

Periodically, the AT Scheduler, described in Section 4.4.1, will decide to change the AT currently in use as a mechanism for combating tracking attacks. Before selecting a new AT, the PKI Client will check whether the AT is about to be selected to sign messages will be considered as false positive by the receiver. In the case of this occurrence, the PKI Client will discard it and proceed to use a backup one. This verification process will be done by using the bloom filter/s included in the CRL received.

[1] Rekey procedure is the procedure of assigning a new EC to a vehicle starting from an old EC.

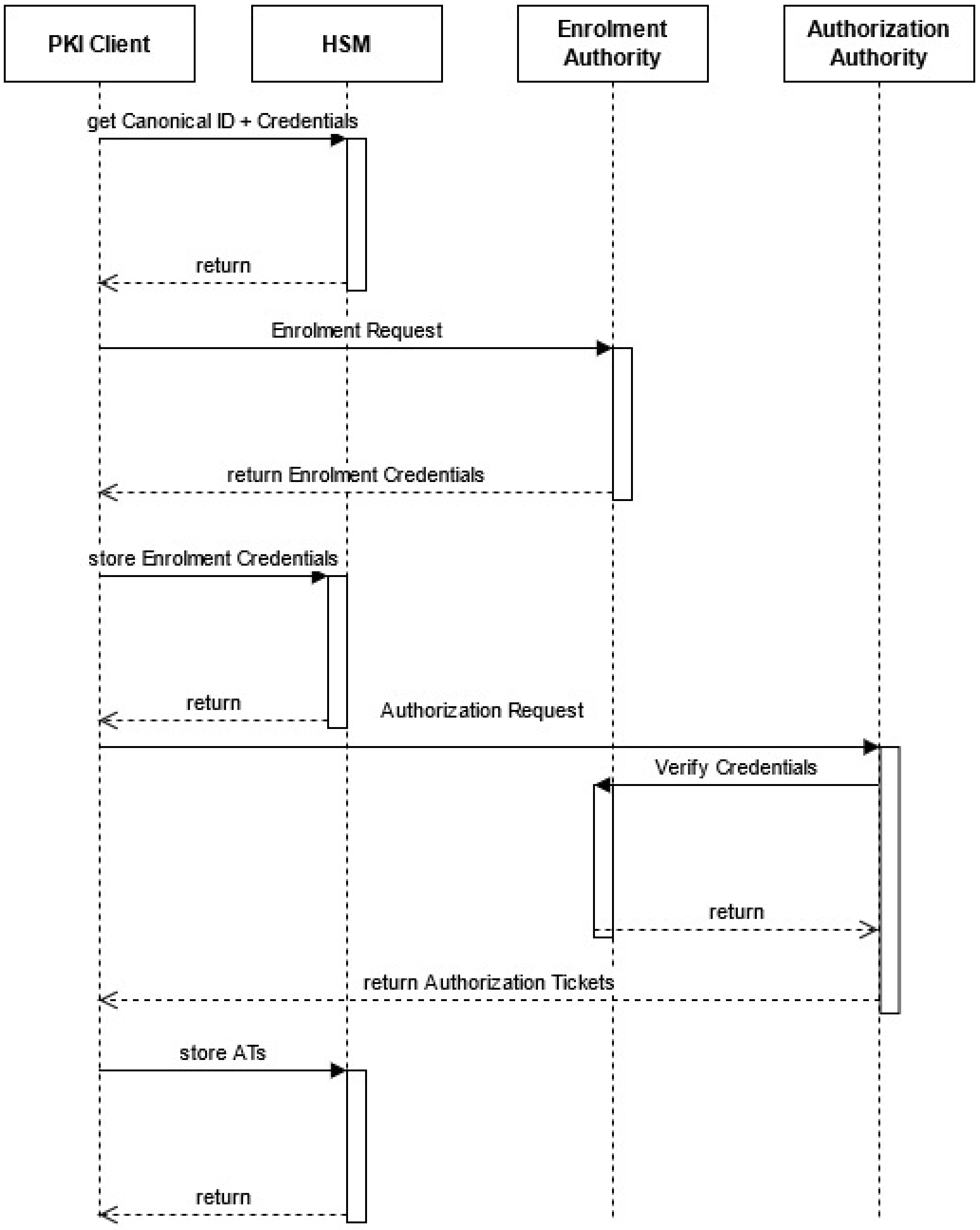

Figure 28: PKI Client Sequence Diagram

5.3 Interfaces Model

This section summarizes the details of the interfaces exposed by the core PKI Toolbox components, as presented in Table 2.

Component	Interface	Type	Description
Root CA Service	Create Online CA	admin	Creation of the certificates for the subordinate certification authorities
	Create Enrolment CA		
	Create Authorization CA		
	Create Revocation CA		
Enrolment Service	Enrolment Request	API REST	INPUT: Canonical ID+ OUTPUT: Enrolment credentials
	Enrolment Management	admin	Management of ITS-Ss, including its registration, status and permissions
	Enrolment Renewal	API REST	INPUT: Canonical ID + Bootstrap Certificate OUTPUT: Enrolment Credentials
	Enrolment Verification	API REST	INPUT: Enrolment Credentials OUTPUT: Valid (Y/N)
Authorization Service	Authorization Request	API REST	INPUT: Enrolment Credentials OUTPUT: Authorization Tickets
Revocation Service	Revocation Request	API REST	INPUT: Certificate OUTPUT: Success (Y/N)
	Revocation Status	API REST	INPUT: Certificate OUTPUT: Valid (Y/N)
MEC Revocation Service	Get CRL Update	API REST	INPUT: CRL OUTPUT: Success (Y/N)
	Current CRL	MQTT	Broadcasts the current CRL version

	Get CRL	MQTT	Broadcasts the current CRL

Table 2: Interfaces of the PKI Toolbox Components

6 Prototype Implementation

The PKI Toolbox has been implemented following a microservice architecture with the objective of increasing the flexibility, scalability and resilience of the prototype. Microservices represent an architectural framework applicable to large-scale projects that allow developers to address separately the development and deployment processes of the services. Microservices are independent units, meant to handle concrete business logic functions, that are combined with database units to form a complete application.

The PKI Toolbox microservices have been developed in Java by using the Spring Boot framework [51]. Spring Boot is an open-source framework which helps you to build production ready and stand-alone spring applications by providing default annotation and codes configuration. Spring Boot also helps in the integration with the Spring ecosystem which includes Spring Data, Spring Security, Spring ORM, and Spring JDBC.

6.1 Prerequisites and Installation

The PKI Toolbox will require the following software components for a successful installation and operation:

- JDK 1.8 or later
- Gradle 4+ or Maven 3.2+
- Spring Framework 5.3.3+
- Spring Boot 2.4.2+

The first step should be to proceed with the installation of the Spring Boot which can be used in the same way as any other standard Java library, simply including the appropriate *spring-boot-*.jar* files on your classpath. However, we strongly recommend the use of a build tool that supports the dependency management, such as Gradle or Maven. Spring Boot is compatible with both.

Download and unzip the source repository for this guide, or clone it using Git:

git clone https://github.com/spring-guides/gs-spring-boot.git

6.2 Source Code Repository

The source code of the PKI Toolbox will be uploaded in an artefact repository managed by the CARAMEL consortium. It will be available only for authorized internal use by the CARAMEL consortium within the scope of the project.

7 Innovation Summary

The innovations provided by the PKI Toolbox defined in the context of CARAMEL Task 4.1 can be summarized as follows:

- A Machine Learning algorithm has been designed and trained to track vehicles from the V2X messages sent by those vehicles. The training has been carried out with an extremely large dataset of realistic synthetic data. This ML approach differs from the classical ruled based algorithms.
- The proposed AT scheduler identifies the best moment to change the AT of the car in order to deceive a potential tracking algorithm. While conventional AT schedulers have a fixed interval to make these changes, the proposed decision engine makes a dynamic scheduling by evaluating how easily it is to track the car at each time step. In this way, the system takes into account the dynamic surrounding conditions of traffic.
- The system tackles the V2X counter-tracking task in a way that is similar to an Adversarial Machine Learning (AML) problem, which is a technique that attempts to cause a malfunction in a ML algorithm by introducing some type of deceptive inputs.
- The proposed approach reduces the size of the Certificate Revocation List (CRL), compressing it through the implementation of Bloom Filters. This technique yields smaller CRL which can be transmitted faster to the vehicles, reducing the window of opportunity for malicious vehicles to attack.
- A microservices architecture ensures the scalability of the PKI Toolbox and provides the required level of flexibility to adapt the service instances the workload changes or deployment requirements.

8 Conclusions

This document describes the work done in the context of Task 4.1 of CARAMEL project, aiming to define a PKI-based Identity Management System for Vehicles, fully compliant with ETSI standards, and addressing the specific security and privacy requirements that should be fulfilled in VATNETs to deal with cyber-threats. Besides this, VANETs also bring in new challenges and conflicts between these security and privacy requirements that were carefully addressed in this activity. More specifically, we addressed anonymity and scalability that are fundamental requirements for the PKI adoption in VANETs.

Although the use of short-term identities can guarantee identity privacy, they cannot ensure location privacy. If a vehicle changes its identity certificate between two observation points controlled by an attacker while moving in the same lane and with the same speed on the road, an attacker can correlate the certificates used by that vehicle and hence track the vehicle. In this context, we addressed this issue by defining an effective AT scheduler architecture (see Section 4.4.1) that decides the best moment to change the AT of a target vehicle to avoid being tracked by an attacker.

The unlinkability of short-term identities guarantees anonymity but on the other hand brings new challenges related to the management of the revoked certificates and their timely distribution. The prompt revocation of misbehaving or malfunctioning vehicles is a crucial point in the PKI Toolbox design, to prevent malicious vehicles from communicating with their surroundings and potentially take advantage of the infrastructure. Consequently, some optimization approaches were evaluated in Section 4.1 and a mechanism for ensuring its timely distribution was defined based on bloom filters.

To complement this work, Section 5 provides a detailed view on the PKI Toolbox architecture, including the description of the main components and its interfaces. While Section 6 closes its definition by including details about the prototype implementation and deployment.

9 References

[1] ETSI TS 102 941 - V1.3.1 - Intelligent Transport Systems (ITS); Security; Trust and Privacy Management. ETSI, 2019

[2] ETSI TS 102 940 - V1.3.1 - Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management. ETSI, 2018.

[3] "A brief history of modern communication security and PKI | Nexus Group", Nexusgroup. [Online]. Available: https://www.nexusgroup.com/a-brief-history-of-modern-communication-security-%E2%80%AFand-why-pki-is-the-state-of-the-art/ [Accessed: 2021].

[4] R. Prichard, History of Encryption. SANS - GIAC, 2002.

[5] K. Richards, "What is cryptography?", SearchSecurity, 2020. [Online]. Available: https://searchsecurity.techtarget.com/definition/cryptography#:~:text=Cryptography%20is%20a%20method%20of,%22%20stands%20for%20%22writing.%22 [Accessed: 2021].

[6] "What is PKI? A Public Key Infrastructure Definitive Guide - Keyfactor", Info.keyfactor.com. [Online]. Available: https://info.keyfactor.com/what-is-pki#what_is_pki. [Accessed: 2021]

[7] S. Haines, A. Byrne, B. Ammar, L. O'Sullivan, A. Waller and T. Oder, Secure Architectures of Future Emerging cryptography. European Commission, 2016.

[8] "Components of a PKI", Ncsc.gov.uk, 2020. [Online]. Available: https://www.ncsc.gov.uk/collection/in-house-public-key-infrastructure/introduction-to-public-key-infrastructure/components-of-a-pki. [Accessed: 2021].

[9] S. Mazaher and P. Røe, A Survey of State of the Art in Public Key Infrastructure. Norsk Regnesentral, 2003.

[10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, "RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Tools.ietf.org, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280.html. [Accessed: 2021].

[11] R. Prodanović, I. Vulić and I. Tot, A SURVEY OF PKI ARCHITECTURE. Association of Economists and Managers of the Balkans, 2019.

[12] A. Albarqi, E. Alzaid, F. Ghamdi, S. Asiri and J. Kar, Public Key Infrastructure: A Survey. Journal of Information Security, 06. 31-37. 10.4236/jis.2015.61004. 2015.

[13] "How PGP works", Users.ece.cmu.edu. [Online]. Available: https://users.ece.cmu.edu/~adrian/630-f04/PGP-intro.html#:~:text=PGP%20is%20a%20hybrid%20cryptosystem,plaintext%20to%20crack%20the%20cipher. [Accessed: 2021].

[14] B. Wolford, "What is PGP encryption and how does it work?", ProtonMail Blog, 2019. [Online]. Available: https://protonmail.com/blog/what-is-pgp-encryption/. [Accessed: 2021].

[15] "Section 14.3. Public-Key Infrastructure | Cryptography and Network Security (4th Edition)", Flylib.com. [Online]. Available: https://flylib.com/books/en/3.190.1.122/1/. [Accessed: 2021].

[16] "PKI", Oasis-pki.org. [Online]. Available: http://www.oasis-pki.org/resources/techstandards/. [Accessed: 2021].

[17] Y. Wang, Public-Key Cryptography Standards: PKCS. University of North Carolina.

[18] S. Tuecke, V. Welch, D. Engert, L. Pearlman and M. Thompson, "RFC 3820 - Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile", Tools.ietf.org, 2004. [Online]. Available: https://tools.ietf.org/html/rfc3820. [Accessed: 2021].

[19] C. Adams, S. Farrell, T. Kause and T. Mononen, "RFC 4210 - Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", Tools.ietf.org, 2005. [Online]. Available: https://tools.ietf.org/html/rfc4210. [Accessed: 2021].

[20] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, "RFC 6960 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", Tools.ietf.org, 2013. [Online]. Available: https://tools.ietf.org/html/rfc6960. [Accessed: 2021].

[21] S. Chokhani, W. Ford, R. Sabett, C. Merrill and S. Wu, "RFC 3647 - Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", Tools.ietf.org, 2003. [Online]. Available: https://tools.ietf.org/html/rfc3647. [Accessed: 2021].

[22] J. Schaad, "RFC 4211 - Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", Tools.ietf.org, 2005. [Online]. Available: https://tools.ietf.org/html/rfc4211. [Accessed: 2021].

[23] J. Schaad and M. Myers, "RFC 5272 - Certificate Management over CMS (CMC)", Tools.ietf.org, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5272. [Accessed: 2021].

[24] S. Santesson, M. Nystrom and T. Polk, "RFC 3739 - Internet X.509 Public Key Infrastructure: Qualified Certificates Profile", Tools.ietf.org, 2004. [Online]. Available: https://tools.ietf.org/html/rfc3739. [Accessed: 2021].

[25] C. Adams, P. Cain, D. Pinkas and R. Zuccherato, "RFC 3161 - Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", Tools.ietf.org, 2001. [Online]. Available: https://tools.ietf.org/html/rfc3161. [Accessed: 2021].

[26] S. Farrell, R. Housley and S. Turner, "RFC 5755 - An Internet Attribute Certificate Profile for Authorization", Tools.ietf.org, 2010. [Online]. Available: https://tools.ietf.org/html/rfc5755. [Accessed: 2021].

[27] R. Kuhn, V. Hu, W. Polk and S. Chang, "SP 800-32, Introduction to Public Key Technology and the Federal PKI Infrastructure", csrc.nist.gov, 2001. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-32/final. [Accessed: 2021].

[28] W. Burr, D. Dodson, N. Nazario and W. Polk, "SP 800-15, MISPC Minimum Interoperability Specification for PKI Components, Version 1", csrc.nist.gov, 1998. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-15/final. [Accessed: 2021].

[29] ETSI EN 319 411-1 - V1.2.2 - Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates; Part 1: General requirements. ETSI, 2018.

[30] ETSI EN 302 665 - V1.1.1 - Intelligent Transport Systems (ITS); Communications Architecture. ETSI, 2010

[31] ETSI TS 103 097 - V1.3.1 - Intelligent Transport Systems (ITS); Security; Security header and certificate formats. ETSI, 2017.

[32] "ISO 21188:2018 - Public key infrastructure for financial services — Practices and policy framework", https://www.iso.org/, 2018. [Online]. Available: https://www.iso.org/standard/63134.html. [Accessed: 2021].

[33] "ISO 17090-1:2013 - Health informatics — Public key infrastructure — Part 1: Overview of digital certificate services", https://www.iso.org/, 2013. [Online]. Available: https://www.iso.org/standard/63019.html. [Accessed: 2021].

[34] "ISO/TR 21186-3:2021 - Cooperative intelligent transport systems (C-ITS) — Guidelines on the usage of standards — Part 3: Security", https://www.iso.org/, 2021. [Online]. Available: https://www.iso.org/standard/79949.html. [Accessed: 2021].

[35] "ISO/IEC 9594-8:2020 - Information technology — Open systems interconnection — Part 8: The Directory: Public-key and attribute certificate frameworks", https://www.iso.org/, 2020. [Online]. Available: https://www.iso.org/standard/80325.html. [Accessed: 2021]

[36] V. Kumar, J. Petit, and W. Whyte, ‘‘Binary Hash Tree based Certificate Access Management for Connected Vehicles,’’ in Proceedings of the 10th ACM Conference on Security & Privacy in Wireless and Mobile Networks (ACM WiSec), Boston, USA, July 2017

[37] M. A. Simplicio Jr, E. L. Cominetti, H. K. Patil, J. E. Ricardini, and M. V. M. Silva, ‘‘ACPC: Efficient Revocation of Pseudonym Certificates using Activation Codes,’’ Elsevier Ad Hoc Networks, July 2018.

[38] K. Rabieh, M. M. Mahmoud, K. Akkaya, and S. Tonyali, ``Scalable certificate revocation schemes for smart grid AMI networks using bloom filters,'' IEEE Trans. Dependable Secure Comput., vol. 14, no. 4, pp. 420-432, Jul. 2017.

[39] G. Rigazzi, A. Tassi, R. J. Piechocki, T. Tryfonas, and A. Nix, ``Optimized certificate revocation list distribution for secure V2X communications,'' in Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall), Sep. 2017, pp. 1-7.

[40] Kumar, Virendra & Petit, Jonathan & Whyte, William. (2017). Binary hash tree-based certificate access management for connected vehicles. 145-155. 10.1145/3098243.3098257.

[41] Simplicio, Marcos & Cominetti, Eduardo & Kupwade Patil, Harsh & Ricardini, Jefferson & Silva, Marcos. (2018). ACPC: Efficient revocation of pseudonym certificates using activation codes. Ad Hoc Networks. 90. 10.1016/j.adhoc.2018.07.007.

[42] K. M. Tuladhar and K. Lim, ``Efficient and scalable certificate revocation list distribution in hierarchical VANETs,'' in Proc. IEEE Int. Conf. Electro/Inf. Technol. (EIT), May 2018, pp. 0620-0625.

[43] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[44] S. Uppoor, O. Trullols-Cruces, M. Fiore, J.M. Barcelo-Ordinas, “Generation and Analysis of a Large-scale Urban Vehicular Mobility Dataset”, IEEE Transactions on Mobile Computing, Vol.13, No.5, May 2014

[45] Theodore P. Hill. “Knowing When to Stop”. American Scientist, vol. 97, no. 2, March-April 2009, p. 126. DOI: 10.1511/2009.77.126

[46] https://cloud.spring.io/spring-cloud-config/reference/html/#_spring_cloud_config_server

[47] https://github.com/Netflix/eureka

[48] https://www.consul.io/

[49] https://cloud.spring.io/spring-cloud-gateway/reference/html/

[50] https://github.com/Netflix/zuul/wiki

[51] https://spring.io/projects/spring-boot