Implementing a security layer on an ITS simulator

Mathias Kautz
(supervisor: Badis Hammi)

Technical Report n^o202306-techrep-KAUTZ, June 2024
revision d8fe81f

The development and testing of new applications in Cooperative Intelligent Transportation Systems (C-ITS) environ- ments, which rely on Vehicle-to-Everything (V2X) communica- tion, is frequently supported through simulations. Nevertheless, most of existing simulators are either outdated or do not consider the latest adopted standards. Especially, the security and privacy mechanisms of vehicles and V2X communications. In this context, we introduce Platelet, which, to the best of our knowledge, stands as the first V2X simulator compliant with security and privacy standards.

Le développement et le test de nouvelles applications dans les environnements de systèmes de transport intelligents coopératifs (C-ITS), qui reposent sur la communication Véhicule-à-Tout (V2X), sont souvent soutenus par des simulations. Néanmoins, la plupart des simulateurs existants sont soit obsolètes, soit ne prennent pas en compte les dernières normes adoptées, en particulier les mécanismes de sécurité et de confidentialité des véhicules et des communications V2X. Dans ce contexte, nous présentons Platelet qui, à notre connaissance, est le premier simulateur V2X conforme aux normes de sécurité et de confidentialité.

Keywords
ITS, ETSI, CAM, security, signature

Laboratoire de Recherche de l'EPITA
14-16, rue Voltaire – FR-94276 Le Kremlin-Bicêtre CEDEX – France
Tél. +33 1 53 14 59 22 – Fax. +33 1 53 14 59 13
mathias.kautz@epita.fr – http://www.lre.epita.fr/

Copying this document

Contents

Chapter 1

Introduction

In the context of modern smart cities, Cooperative Intelligent Transportation Systems (C-ITS) represent one of the main use cases that aim to improve citizens' daily lives (Hammi et al., 2022). C-ITS technologies strive to increase road safety, efficiency, and comfort by integrating the processes of sensing, communication, decision-making, and acting based on the surrounding road environment. Consequently, a multitude of communication types are intricately involved in a C-ITS environment, commonly referred to as Vehicle-to-Everything (V2X) communications. At the beginning, these networks were designed to be ad-hoc, but more recent architectures have proved that some kind of infrastructure is very handy to connect centralized systems to the network (Lee and Atikson, 2021). C-ITS is now primarily composed of two types of nodes: Intelligent Transport System Station-Vehicle (ITSS-V) and Intelligent Transport System Station-Roadside Unit (ITSS-R). For example, ITSS-R nodes can be used to retrieve certificates from a PKI (Public Key Infrastructure) hosted on the internet.

In vehicular networks, topology changes very rapidly and there is frequent signal disruption because of the relative speed between nodes. Thus, data delivery is one of the most difficult challenges developers face. C-ITS networks allow the exchange of information that can increase driver and sensor line of sight, making roads safer.

It also provides contextual information to the vehicles nearby and improves the general awareness of all vehicles. C-ITS components communicate using various standards and protocols that define properties like bandwidth or communication range (Anwer and Guy, 2014).

Many different standards exist for C-ITS (IEEE, ETSI). However, this paper will focus only on the ETSI (European Telecommunications Standards Institute) standard. In these systems, security features are very important; a bug or security exploit can be quite literally deadly. The solution chosen by the ETSI to implement such security features is to use a PKI (Public Key Infrastructure) (ETSI, 2021). PKIs are infrastructures that host a set of tools and services to manage the lifecycle of digital certificates. PKIs provide four fundamental guarantees:

- Confidentiality: Assurance that no entity can maliciously or unwittingly view a payload in clear text. Data is encrypted to make it secret, such that even if it was read, it appears as gibberish

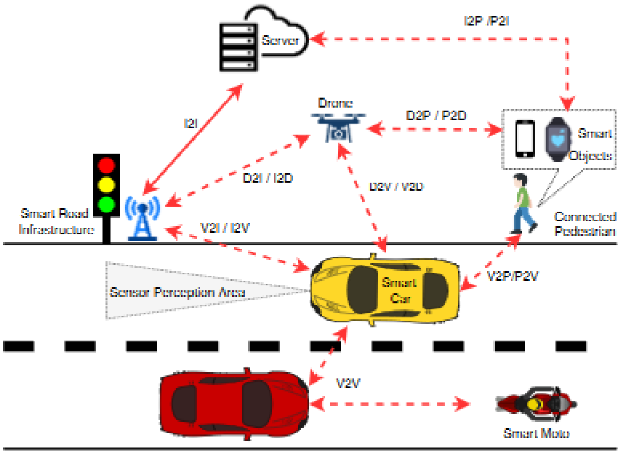


Figure 1.1: C-ITS example (Hammi et al., 2022)

- Integrity: Assurance that if an entity changed (tampered) with transmitted data in the slightest way, it would be obvious it happened as its integrity would have been compromised.
- Authenticity: Assurance that every entity has certainty of what it is connecting to, or can evidence its legitimacy when connecting to a protected service.
- Non repudiation: Assurance that a user of the network cannot refute that he has sent a message using his certificate.

PKI architectures are often composed of different types of authorities issuing different types of certificates serving different kinds of purposes. For example, the generic ETSI PKI implementation proposes three kinds of authorities (Hammi et al., 2022).

- Root Certification Authority (RCA) : the root of all trust inside the PKI. The root CA provides a self signed certificate that is used to sign all other certificates. RCAs define common policies among all subordinate LTCAs and PCAs.
- Long Term Certificate Authority (LTCA) : this authority provides a long term certicate. The LTCs are valid for longer period of time and are used to identify and authenticate the respective ITSS inside the PKI
- Pseudonym Certificate Authority (PCA) : issues pseudonym certificate. This is the certificate used for V2X (vehicule to anything) communication. It has a short lifetime and it contains minimal information to preserve the privacy of the user.

Company cannot test their C-ITS solutions in real-life every time: it would be far too costly and complicated. To mitigate costs, they can program their own network simulations. Many simulators exist, some realistic, some simple, some fast, some slow, some open source, some closed source. This paper will be focused on the extension of the Artery simulator.

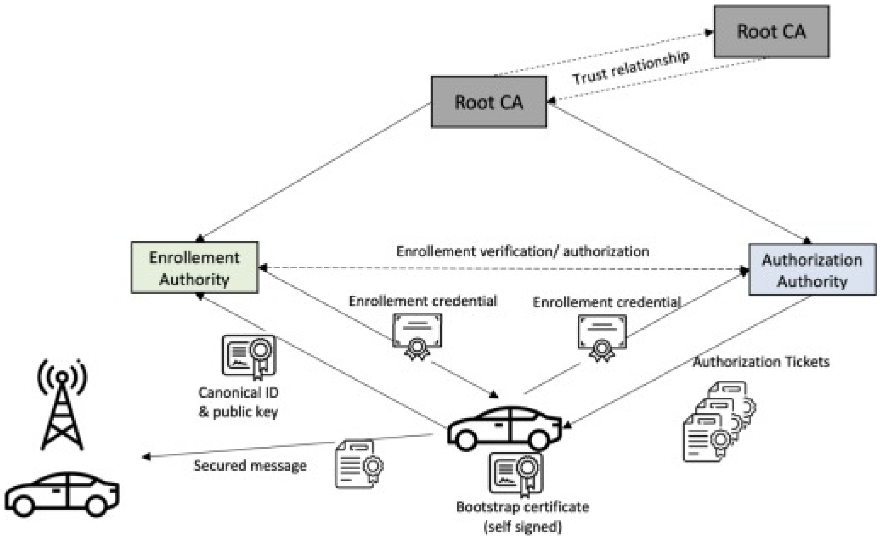


Figure 1.2: ETSI generic PKI (ETSI, 2021)

Chapter 2

State of the art

In this section we will provide a concise tour of the most used C-ITS simulators and explain why we decided to work with Artery

2.1 Traffic and Network Simulation Environment (TraNS)

The Traffic and Network Simulation Environment (TraNS) (Piorkowski et al., 2008) stands out as the oldest simulation platform for Vehicular Ad-Hoc Networks (VANET). Built upon the NS2 network simulator and the SUMO traffic simulator, TraNS pioneered a realistic simulation approach for VANETs to mitigate the significant discrepancies between simulation results and real-world experiments. Despite its historical significance, TraNS, rooted in the outdated NS2 network simulator, faces limitations in supporting large-scale simulations and accurately modeling VANET protocols. However, security and privacy mechanisms for vehicles and V2X communications have not been incorporated.

2.2 GrooveSim

GrooveSim (Mangharam et al., 2005) represents another older VANET simulator. Unlike the other simulators discussed in this section, GrooveSim does not rely on existing traffic or network simulators. Instead, its primary objective is to intricately model inter-vehicular communication within a real street map-based topography. GrooveSim is designed to comply with the 802.11p/1609 C-ITS communication protocols. Notably, it represents a hybrid simulator, facilitating interaction between both real and virtual vehicles in the simulation environment. Nonetheless, the security and privacy mechanisms for vehicles and V2X communications are not implemented.

2.3 Vehicles in Network Simulation (Veins)

iTETRIS (Rondinonea et al., 2013), is an EU-funded simulator. It represents an extension of TraNS and uses SUMO as a traffic simulator but distinguishes itself by upgrading from NS2 to NS3, a network simulator capable of accurately simulating a large number of nodes. The development of the iTETRIS simulation platform, funded by the EU, was motivated by the lack of simulation platforms capable of precisely modeling and testing C-ITS in expansive scenarios.

Table 2.1: Comparison of VANET simulators

Simulator	Traffic simulation	Network simulation	Hybrid	ETSI-compliant	IEEE-compliant	Signature implementation	Certificate renewal	pcap logging	EC and PC implementation	Certificate pool implementation
TraNS	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
GrooveSim	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗
iTETRIS	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗
Veins	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
Artery	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗
PLATELET	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓

As an EU project, the iTETRIS simulator adheres to the ITS-G5 cooperative ITS communication protocols standard. However, like the previously described simulators, it does not implement security and privacy mechanisms for vehicles and V2X communications.

2.4 Veins

The Vehicle In Network Simulation (Veins) framework Sommer (2021) represents a contemporary and comprehensive simulation framework built upon SUMO and Omnet++. Veins is designed to be a swift, user-friendly, and highly adaptable platform. It seamlessly incorporates the IEEE 802.11p/1609 cooperative ITS standard, while also offering extensions, such as Artery, to support ETSI ITS-G5. Veins stands out as the contemporary choice for developing simulations of C-ITS when using the IEEE 1609 and WAVE cooperative standards. Yet, it does not support the security and privacy mechanisms for vehicles and V2X communications.

2.5 Artery

Artery (Riebl et al., 2019) represents an extension of Veins, that is specifically crafted to implement the ETSI ITS-G5 C-ITS communication protocol standard while seamlessly integrating withthe Veins framework. It relies on Vanetza [19], an open-source implementation of the ITS-G5 standard, which incorporates essential features for simulating a network based on the ETSI specifications, including GeoNetworking, and Broadcast Technical Protocol (BTP) protocols. Artery, implements few security functions in its source code. Still, no security function is implemented for simulation scenarios. Security and privacy management are crucial to all C-ITS simulation scenarios and it is more important in privacy aware applications like IDS. However, as Table I highlights none of the most used VANET simulators implements security management.

2.6 Conclusion

After this quick tour of existing simulator it becomes clear that Artery is the best choice to implement a security layer based on the ETSI standard. It is a fast, modern and open source simulator that already implements the European standard through Vanetza.

Chapter 3

Artery

In this chapter we will take a look at the Artery architecture; what's already implemented and what's lacking inside the simulator. Like said before Artery is based on two major components: it's traffic simulator SUMO and its network simulator Omnet++. Those two components are synced using a TCP based client/server bidirectional connection named TraCI.

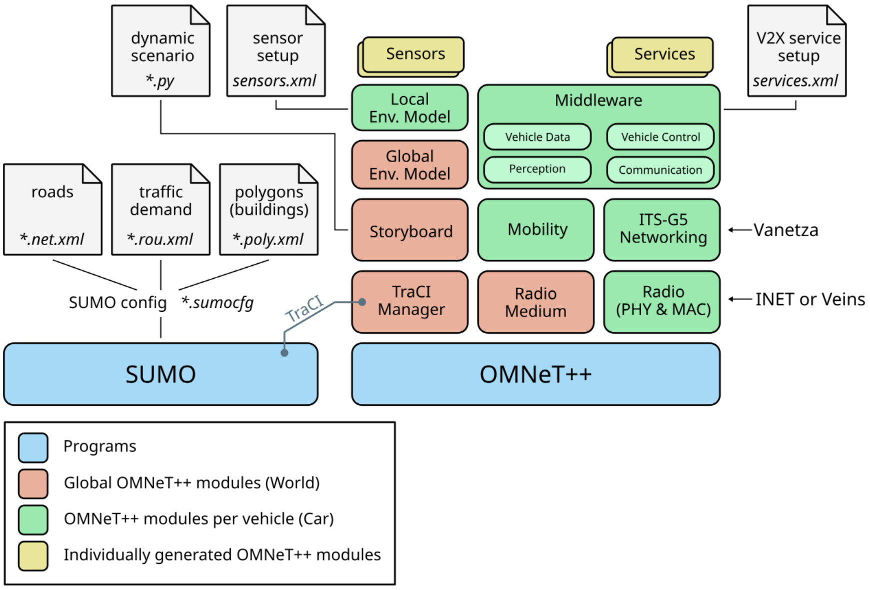


Figure 3.1: Artery architecure (Riebl, 2015)

3.1 SUMO

"Simulation of Urban MObility" (SUMO) is an open-source, highly portable, microscopic, and continuous traffic simulation package designed to handle large networks (xobx cherif, 2017). It allows for intermodal simulation, including pedestrians, and comes with a large set of tools for scenario creation. It is mainly developed by employees of the Institute of Transportation

Systems at the German Aerospace Center. With SUMO, you can put together simulation scenarios using XML files describing vehicle behavior. You can also import OpenStreetMap (OSM) chunks to run your simulation on real-life traffic topology. In this paper, we will be using this workflow:

1. Export OSM chunk from internet and convert it to netfile using the netconvert utility.
2. Use the randomTrips.py python script to compute randomly generated vehicule moving across the net.
3. Generate route file with the duarouter tool. The routes are based on the trip file we generated in the previous step.
4. We can now write the sumo.cfg file to parameter our scenario.

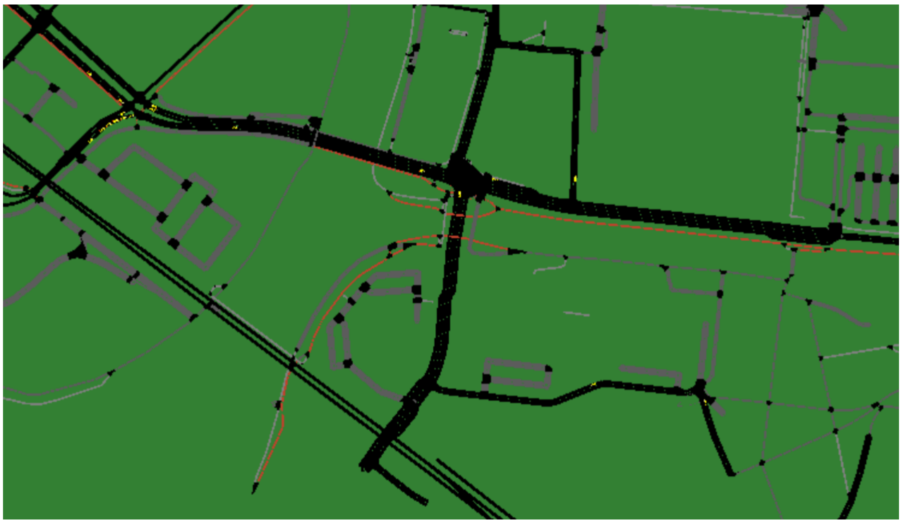


Figure 3.2: An example of SUMO simulation around Epita Strasbourg

3.2 Omnet++

Omnet++ is not a network simulator itself; it is an extensible, modular, component-based C++ simulation library and framework (Omnet++, 2023). Although Omnet++ is capable of more than network simulation, in this paper, we will use it as our network simulator. Omnet++ is based on the principle of components (or modules) written in C++ and assembled into larger components using the NED topology description language. Components are programmed in C++ and assembled into larger components and models in a high-level language called NED. On top of Omnet++, Artery comes with INET (INET, 2023). INET is an open-source model for wired, wireless, and mobile networks. Since VANET is based on the IEEE 802.11p IPv6 and TCP/UDP stack (Rehman et al., 2013), we can implement it using INET. Most of Artery's value comes from the new Omnet++ modules it provides, like the Middleware, the Router, etc.

3.3 Vanetza

Vanetza (Riebl et al., 2017) is a stand-alone implementation of various components of an ETSI ITS-G5 protocol stack. It includes GeoNetworking, BTP, and DCC (ETSI, 2013) (ETSI, 2012). While secured packet structures, classes, and methods are partially implemented inside Vanetza (ETSI, 2016), they are not yet implemented inside Artery. Implementing these components in Artery is part of the work presented in this paper.

3.4 Artery Architecture

As you can see in the architecture Artery is composed of many subsystems and components like Vanetza, INET, etc. In this section, I will talk about important Omnet++ component inside the artery framework.

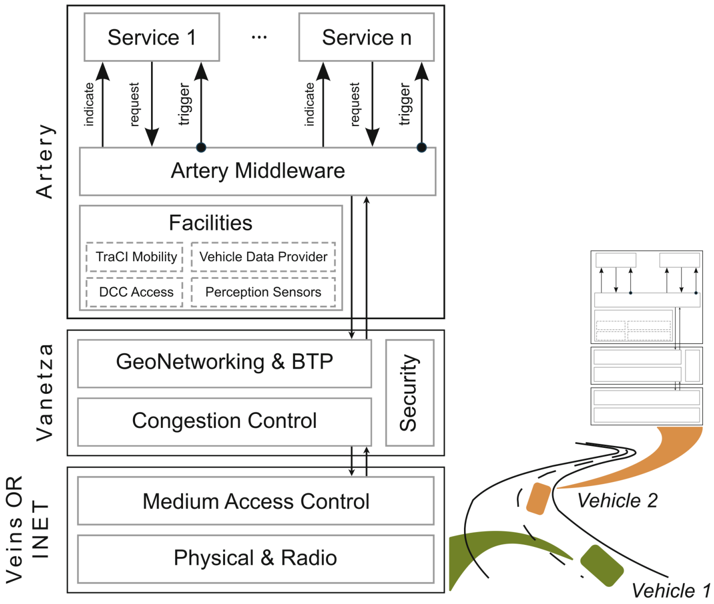


Figure 3.3: Artery full architecure (Riebl et al., 2019)

3.4.1 ITS-G5 Middleware

The Middleware is the key component of the Artery simulation framework (Riebl et al., 2019). It serves as the backbone of all Artery services, acting as an information hub and providing interfaces to further components. Upon initialization, the Middleware also initializes the GeoNetworking router. Additionally, the Middleware is responsible for routing messages to and from node services.

3.4.2 ITS-G5 Services

Inside Artery, a service is a stand-alone application of a vehicle. Within the framework, a service is simply a C++ class derived from the ITSG5BaseService class. ITSG5BaseService itself is derived from the Omnet++ class hierarchy, thus providing all Omnet++ functionality to services such as handling events and collecting statistics. Each service can send messages defined using ASN.1 syntax or Omnet++ cPacket. To interact with the simulation, a service has to implement three important methods:

- Initialize: this method is called once during the instanciation of the service by the middleware. It is often used to set constant or construct objects used later in the simulation.
- Trigger: this method is called periodically by Artery (the period can be specified in artery configuration). Trigger is used to send periodic message (CAM for example) to other vehicules or to perform general periodic tasks.
- Indicate: this method is called whenever the vehicle receives a message. When a message is received, the service needs to verify the integrity of the message (for example by checking checking every piece of information is within a range of acceptable values). If the packet is valid, the service consumes it and updates informations according to the message type.

One exemple service provided by Artery is the CaService charged to send and consume Cooperative Awereness Messages (CAMs) periodically.

Chapter 4

Platelet

In this paper we present Platelet: an extension to the Artery simulation framework that aims to create simulation scenario that implement security from the ETSI. Platelet is a bundle of Artery's extension, scripts and cli tools and a standalone application allowing users to easily manage the lifecycle of secured scenarios.

4.1 Simulator extensions

While Artery already implements secured message creation, verification, and consumption within Vanetza to make it a ETSI compliant secured simulation framework we needed to implement some critical component such as certificate loading or an after-simulation verification tool. To close the gap, we decided to implement a StaticCertificateProvider that verify and load certificate at the start of the simulation and a PcapItsRecorder that plugs to the router of a node and dump received/send ITS packet to a pcap file.

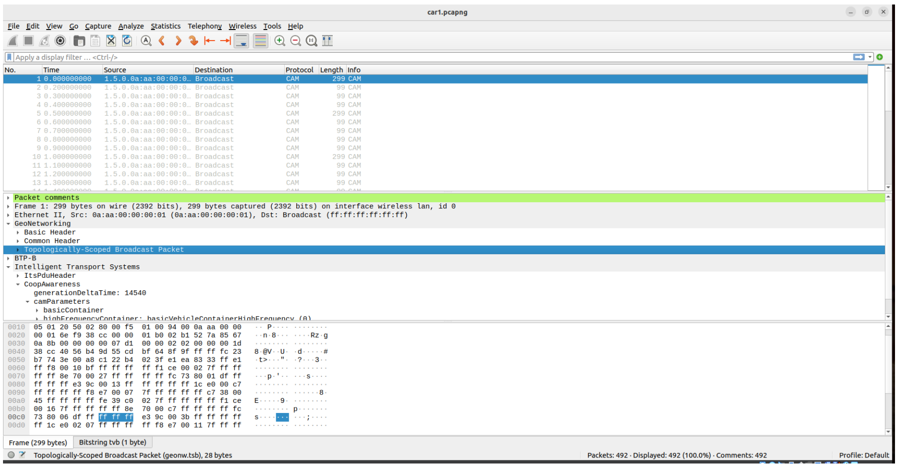
4.1.1 Certificate loader

As we saw earlier ETSI chose a PKI-based security Architecture. With 3 authority: the root authority, the long term certification authority and the pseudonym authority. While Certificate and SecuredMessage data structures exists within Vanetza Artery doesn't implement a proper way to provide those certificate to a network's node that needs it. To allow vehicule to request certificates during the simulation we developed the StaticCertificateProvider. Before the simulation we have to generate every certificate as a file. We can create the complete authority architecture using the certify tool or directly inside the Platelet app. After generating all the certificates Platelet will distribute them during the simulation until it is finished or there is no more valid certificate (therefor we need to generate more to finish the simulation). When the StaticCertificateProvider loads the certificates it verify there integrity and validity so that a broken certificate could never be distributed to a network's node.

4.1.2 Pcap recorder

To verify certificate validity after the simulation ends, it's essential to find a method to save them. The standard format for storing network packets is pcap. Pcap is a lightweight format that can be easily read by software like Wireshark, providing a straightforward interface for inspection.

To facilitate this in Artery, I developed the PcapItsRecorder module. This module is directly integrated into each node of the simulation. It records every message in the configured direction on a specified interface and dumps its contents into a pcap file. The interface name, packet direction (whether sent or received), and output filename can all be configured within the omnetpp.ini file of the scenario. This setup ensures that all network activity, including certificate exchanges, can be captured and later analyzed using standard tools like Wireshark.



```
No.   Time         Source                 Destination  Protocol  Length Info
    1 0.000000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM          299 CAM
    2 0.200000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
    3 0.300000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
    4 0.400000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
    5 0.500000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM          299 CAM
    6 0.600000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
    7 0.700000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
    8 0.800000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
    9 0.900000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
   10 1.000000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM          299 CAM
   11 1.100000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
   12 1.200000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
   13 1.300000000  1.5.0.0a:aa:00:00:0…   Broadcast    CAM           99 CAM
```

```
› Packet comments
› Frame 1: 299 bytes on wire (2392 bits), 299 bytes captured (2392 bits) on interface wireless lan, id 0
› Ethernet II, Src: 0a:aa:00:00:00:01 (0a:aa:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⌄ GeoNetworking
  › Basic Header
  › Common Header
  › Topologically-Scoped Broadcast Packet
› BTP-B
⌄ Intelligent Transport Systems
  › ItsPduHeader
  ⌄ CoopAwareness
      generationDeltaTime: 14540
    ⌄ camParameters
      › basicContainer
      › highFrequencyContainer: basicVehicleContainerHighFrequency (0)
```

```
0010  05 01 20 50 02 80 00 f5  01 00 94 00 0a aa 00 00   ·· P····  ········
0020  00 01 6e f9 38 cc 00 00  01 b0 02 b1 52 7a 85 67   ··n·8···  ····Rz·g
0030  0a 8b 00 00 00 00 07 d1  00 00 02 02 00 00 00 1d   ········  ········
0040  38 cc 40 56 b4 9d 55 cd  bf 64 8f 9f ff ff fc 23   8·@V··U·  ·d·····#
0050  b7 74 3e 00 a8 c1 22 b4  02 3f e1 ea 83 33 ff e1   ·t>···"·  ·?···3··
0060  ff f8 00 10 bf ff ff ff  ff f1 ce 00 02 7f ff ff   ········  ········
0070  ff ff 8e 70 00 27 ff ff  ff ff fc 73 80 01 df ff   ···p·'··  ···s····
0080  ff ff ff e3 9c 00 13 ff  ff ff ff ff 1c e0 00 c7   ········  ········
0090  ff ff ff ff f8 e7 00 07  7f ff ff ff ff c7 38 00   ········  ······8·
00a0  45 ff ff ff ff fe 39 c0  02 7f ff ff ff ff f1 ce   E·····9·  ········
00b0  00 16 7f ff ff ff ff 8e  70 00 c7 ff ff ff ff fc   ········  p·······
00c0  73 80 06 df ff ff ff ff  e3 9c 00 3b ff ff ff ff   s·······  ···;····
00d0  ff 1c e0 02 07 ff ff ff  ff f8 e7 00 11 7f ff ff   ········  ········
```

Topologically-Scoped Broadcast Packet (geonw.tsb), 28 bytes

Packets: 492 · Displayed: 492 (100.0%) · Comments: 492

Profile: Default

Figure 4.1: Wireshark screen capture

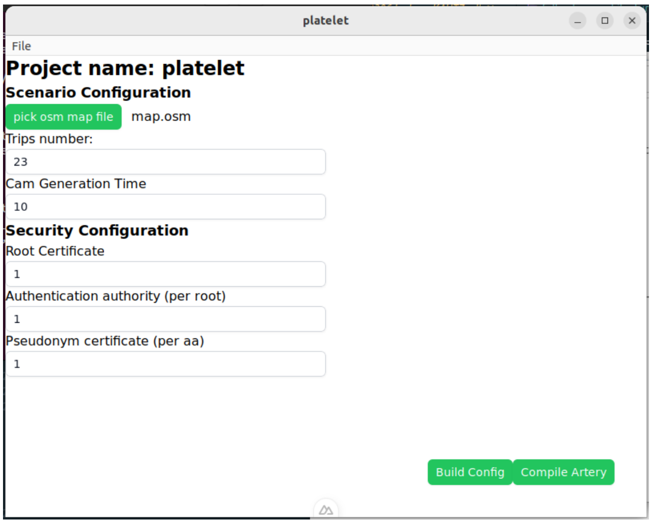
4.2 The Platelet application

In the previous chapter, we explored the various components, tools, utilities, and scripts that Artery requires to function effectively. Each of these tools needs to be set up, adding complexity to the creation of a new secured scenario. To streamline and simplify this process significantly, we have developed Platelet—an intuitive interface for creating secured scenarios within Artery.

Platelet is an application that allows users to create, edit, and delete scenarios through a user-friendly interface. The Platelet app is built using Tauri, with the backend written in Rust and the frontend using Vue3 with Tailwind CSS. In its initial version, the app includes features such as scenario creation, configuration of scenario parameters, and automatic generation of required configurations.

With Platelet, users can easily set up and manage scenarios within Artery without needing deep technical expertise. This simplification enhances the usability of Artery for developing and testing secure communication scenarios in vehicular networks.

We can now click "build configuration" to create all files needed for SUMO, Omnet++, and Artery. It also generates certificates that will be used during the simulation. After the configuration is done, we can click "compile artery" to start the simulation. The Platelet application groups all the tools necessary to create a secured scenario inside a single application.



Project name: platelet

Scenario Configuration

map.osm

Trips number:
23

Cam Generation Time
10

Security Configuration

Root Certificate
1

Authentication authority (per root)
1

Pseudonym certificate (per aa)
1

Figure 4.2: Platelet interface

4.2.1 Documentation and stabilization effort

During the last few month and as a preparation to the NCA2024 we pushed a documentation and stabilization effort where we tried to clean the code base ,fix bugs, create simple documentation and produce example and demos to finish our paper. For example durin this time we created two demos linked below.

4.2.2 Demo

To demonstrate the potential of Platelet we did two videos showing how one can create simply a secured simulation scenario.

The first one is a global demo showing how you can create a default simulation scenario using platelet. This is simulation vehicule exchanging secured cooperative awereness messages around the effeil tower:

https://youtu.be/WuIl59mwxi0

The second one is a simulation of a vehicule performing a Sybil attack. This demo shows the flexibility and customization capabilities of Platelet by allowing people to load custom configuration file and thus apply custom behavior to vehicles:

https://youtu.be/v9YlUluFh-o

Chapter 5

Conclusion

In this paper we introducde Platelet the first simulator allowing the simulation of secured scenario compliant to the ETSI standard. Introducing new component to Artery such as certificate loading and result verification and a standalone App built in Rust and VueJS with Tauri enhancing the experience of people trying to simulation secured ITS scenarios.

5.0.1 Further improvement

As of now Platelet has some limitations. The current biggest one is the naivity of our certificate provider requiring the generation of certificate before the simulation and we have to generate more than what will be consumed during the simulation. We have to implement a RuntimeCertificateProvider that will generate certificate on the run as they are requested. However this implementation is non trivial as a library implementing such behavoir does not exists.

Chapter 6

Bibliography

Anwer, M. S. and Guy, C. (2014). A survey of vanet technologies. (page 4)

ETSI (2012). Etsi ts 102 724 v1.2.1. (page 11)

ETSI (2013). Etsi en 302 636 v1.2.0. (page 11)

ETSI (2016). Etsi ts 103 097 v1.2.1. (page 11)

ETSI (2021). Etsi ts 102 940 v2.1.1. (pages 4 and 6)

Hammi, B., Monteuuis, J.-P., and Petit, J. (2022). Pkis in c-its: Security functions, architectures and projects: A survey. (pages 4 and 5)

INET (2023). Inet documentation. https://omnetpp.org/documentation/. (page 10)

Lee, M. and Atikson, T. (2021). Vanet application past and present and future. (page 4)

Mangharam, R., Weller, D. S., Stancil, D. D., Rajkumar, R., and Parikh, J. S. (2005). Groovesim: a topography-accurate simulator for geographic routing in vehicular networks. (page 7)

Omnet++ (2023). Omnet++ documentation. https://omnetpp.org/documentation/. (page 10)

Piorkowski, M., Raya, M., Lugo, A. L., Papadimitratos, P., Grossglauser, M., and Hubaux, J.-P. (2008). Trans: realistic joint traffic and network simulator for vanets. (page 7)

Rehman, S., Khan, M. A., Zia, T. A., and Khokhar, R. H. (2013). A synopsis of simulation and mobility modeling in vehicular ad-hoc networks (vanets). (page 10)

Riebl, R. (2015). Artery documentation. http://artery.v2x-research.eu/. (page 9)

Riebl, R., Obermaier, C., and Günther, H.-J. (2019). Artery: Large scale simulation environment for its applications. (pages 8 and 11)

Riebl, R., Obermaier, C., Neumeier, S., and Facchi, C. (2017). Vanetza: Boosting research on inter-vehicle communication. (page 11)

Rondinonea, M., Manerosb, J., Krajzewiczc, D., Bauzaa, R., Cataldid, P., Hrizid, F., Gozalveza, J., Kumare, V., Röcklf, M., Line, L., Lazarog, O., Leguayh, J., Haerrid, J., Vazb, S., Lopezh, Y., Sepulcrea, M., Wetterwaldd, M., Blokpoeli, R., and Cartolanoj, F. (2013). itetris: a modular simulation platform for the large scale evaluation of cooperative its applications. (page 7)

Sommer, C. (2006-2021). Veins documentation. https://veins.car2x.org/. (page 8)

xobx cherif (2017). Sumo-openstreetmap. https://github.com/xobx-cherif/Sumo-OpenStreetMap. (page 9)