(ITEA 3 – 15017)

open standard APplication Platform
for carS and TrAnsportation vehiCLEs

Deliverable: D 2.4

"Description and Development of Security Concepts among all Connections"

Work Package: 2

Service Enablers in Intelligent Networks

Task: 2.4

"Development of Secure Data Transmissions and Secure Data Storage"

Document Type:	Deliverable	**Classification:**	Internal
Document Version:	Final	**Contract Start Date:**	01.01.2017
Document Preparation Date:	27.03.2019	**Duration:**	31.12.2019

ITEA 3
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

History

Rev.	Content	Resp. Partner	Date
0.1	Initial document creation and structure	Alexios Lekidis	27.03.2019
0.2	Added initial draft of the Network IDS contribution	Alexios Lekidis	12.04.2019
0.3	Added initial draft of introduction	Zakaria Laaroussi	24.09.2019
0.4	Added Authorization chapter	Sowmya Ravidas	27.09.2019
0.5	Added initial draft of the anomaly detection based machine learning	Matvej Yli-Olli	01.10.2019
0.6	Added Authentication chapter	Zakaria Laaroussi	01.10.2019
0.7	Added conclusion	Zakaria Laaroussi	11.10.2019

Contents

List of Figures

List of Tables

Summary

This deliverable is the fourth deliverable of the APPSTACLE Work Package 2 "Service Enablers in Intelligent Network". It contains the results of T2.4 "Development of Secure Data Transmissions and Secure Data Storage"

1 Introduction

Connected vehicles are encountering exceptional innovative changes, driven by different technology enablers. OEMs are always taking a shot at innovations to improve road safety and driver experience, to spare life and time.

Sadly, security threats are growing making used systems and services easy to exploit. Connected vehicle services require continuous communication with the internet and other entities by deploying V2X protocols.

Since connected vehicles rely on the information provided the network, lack of security gives possibilities to no-authorized entities remote access using different deployed interfaces. Existing cybersecurity methods for the internet may not be suitable for future connected vehicles, therefore new security frameworks must be designed and implemented to protect our vehicles and transport systems. In this deliverable, we addressed some mandatory security aspects, aiming to enable a secure E2E security.
Specifically, this deliverable will address:

- **Network security:** Using two different IDS(Intrusion Detection System)(Statistical based IDS method and Machine learning based IDS) to secure in-vehicular network addressing mainly CAN-BUS protocol.
- **Authorization:** Designing an authorization framework dedicated to ITS(Intelligent transportation system).
- **Authentication:** Developing a driver authentication mechanism ensuring drivers' identity and ownership of a giving vehicle, focusing on two main connectivity scenarios incoverage and out-of-coverage.

2 Network Security

This section provides an overview of the Network Intrusion Detection System (NIDS) that was developed as a network security mechanism for Vehicle-to-Everything (V2X) connectivity. To facilitate the demonstration of the Network IDS we give: 1) an overview of its architecture, functionality and deployment schemes and 2) a realistic set of V2X attacks that was used to evaluate the security services it provides to the APPSTACLE architecture.

2.1 Anomaly Detection - based Statistical Method

2.1.1 Architecture

The architecture of the NIDS is presented in Figure 2.1 and can be also found in Deliverable D1.3 [12]. For V2X communication the Network IDS includes two main modules:

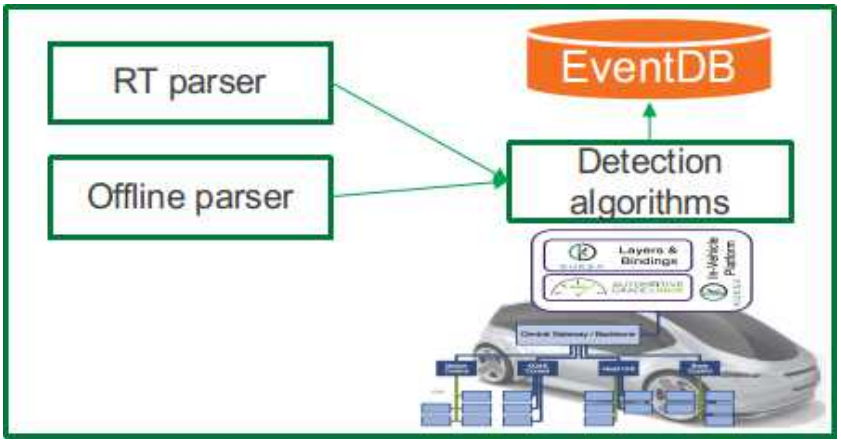


Figure 2.1: NIDS architecture

1. Parser, responsible for capturing and analyzing the packets that are received from the network. The parser includes an encoder/decoder for safety-oriented V2X packets complying to the ETSI EN 302 637-2 [23] and ETSI EN 302 637-3 [24] standards or non-safety V2X packets complying to the TCP/UDP protocol standards. Packet parsing is performed for a certain time interval, called learning phased, that is specified by the user. The module can perform:
   - Real-time analysis of network packets (*RT parser* in Figure 2.1)
   - Analysis through network traffic files i.e. logs, pcaps (*Offline parser* in Figure 2.1)
2. Detection, that includes algorithms to perform anomaly detection on V2X communication. The benefit of this module is that it allows multiple algorithms to be enabled

simultaneously. Once it detects an anomaly, the module generates an alert with all the information to enable incident analysis and mitigation.

The NIDS is integrated in the central gateway. This integration allows it to process network packets from all the vehicle components as the central gateway is responsible for management of the in/ex-vehicle data exhange. The central gateway is positioned in the vehicle as shown in Figure 2.2. It is connected to the Head Unit (vehicle entertainment system), Telematics Control Unit-TCU (vehicle tracking control), On-board Unit-OBU (V2X communication unit with other vehicles/stations), On Board Diagnostics-OBD (vehicle diagnostic port) as well as the internal networks of the vehicle (Deliverable D1.1 [11]). The NIDS placement on the central gateway also allows to detect malicious V2X packets that are received through the OBU before they are mapped to actions that tamper with the in-vehicle architecture.

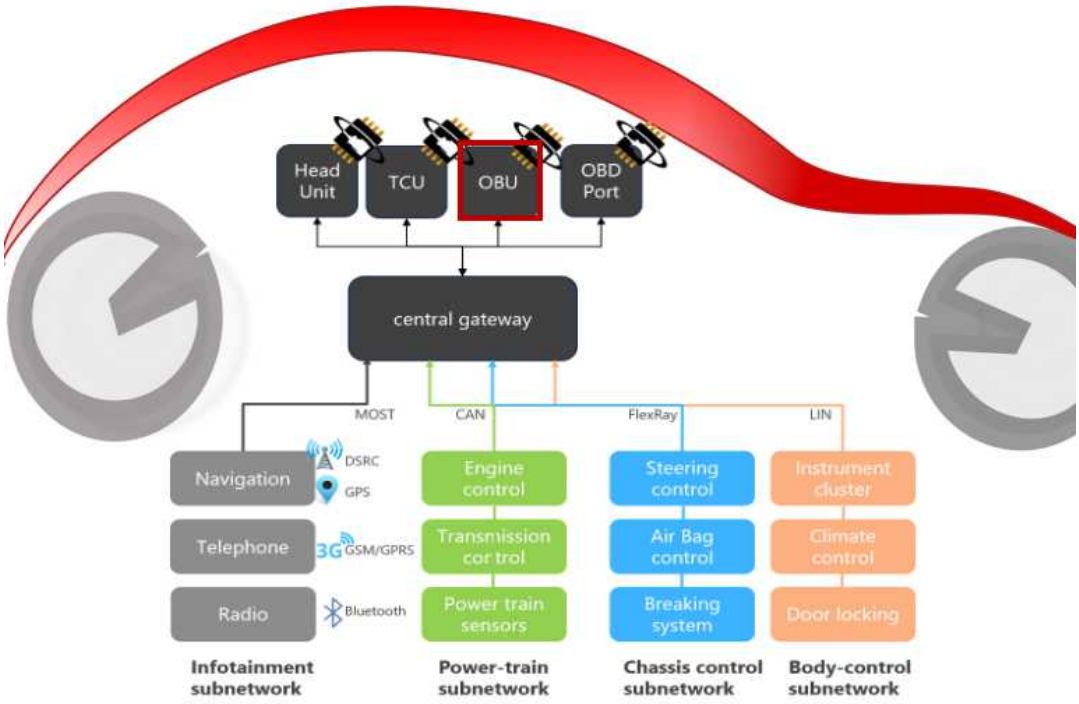


Figure 2.2: Central gateway location in the in-vehicle architecture

2.1.2 Integration with V2X connectivity interfaces

The NIDS is deployed as a software service on the gateway alongside the gateway's network packet logic. The current deployment is shown in Figure 2.3, where the NIDS is integrated as a Linux systemd software service within the APPSTACLE hardware platform. This is performed through a dedicated AGL recipe that is part of the Kuksa in-vehicle platform. The main V2X parsing modules for the NIDS are:

- *ETSI ITS-G5 parser:* This module was developed for handling safety-oriented V2X packets that comply to the ETSI EN 302 637-2/3 standards. Thorough details along with the implemented detection algorithms are described in the associated section.

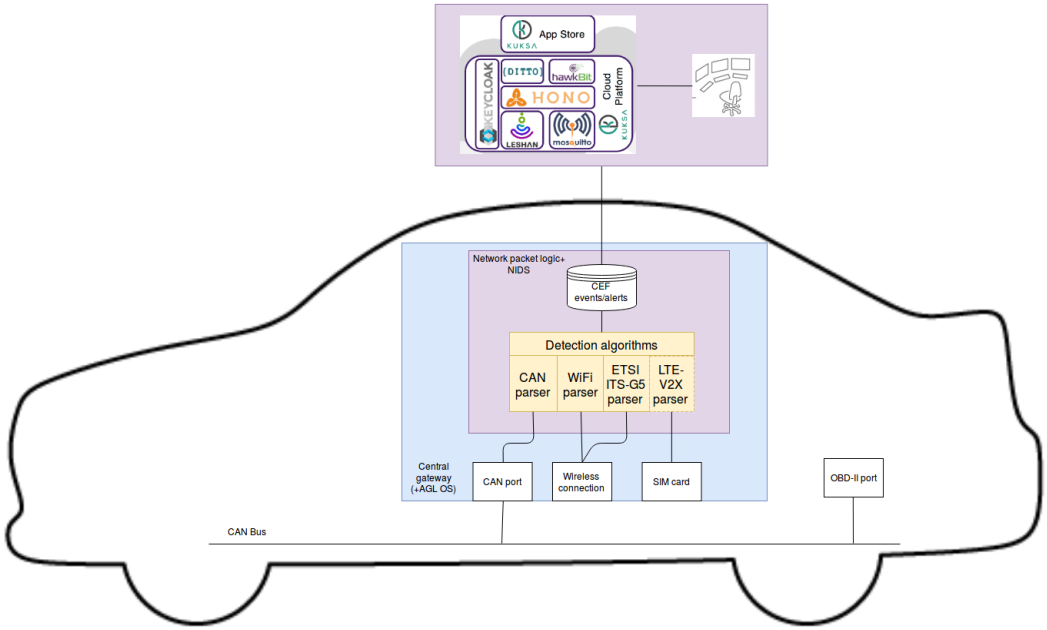


Figure 2.3: NIDS deployment in the central gateway

- *WiFi parser:* This module was developed for handling non-safety-oriented V2X packets that comply to the IEEE 802.11 standard and use TCP or UDP connectivity. As with the previous module, thorough details along with the implemented detection algorithms are described in the associated section.
- *LTE-V2X parser:* This module is planned for parsing the Cellular V2X (C-V2X) messages that are exchanged in V2X connectivity scenarios. Since, C-V2X is currently under standardization process, the implementation of this module has not started yet.

Safety-oriented V2X packets

For the packets complying to ETSI EN 302 637-2/3, the parser module decodes the ASN-1 notation of GeoNetworking, CAM and DENM packets (presented in deliverable D2.2 [14]). Then, it tries to interpret the information inside the packet to build a normal behavior baseline that is stored in a database, called EventDB. This baseline is later used to detect deviations from the normal behavior using the detection module algorithms. The existing algorithms are based on behavior IDS techniques (presented in deliverable D2.1 [13]) or specification IDS techniques [63]. Specifically, the available modules are:

1. *IP whitelist*: Retrieves from the constructed normal behavior baseline the list of IPs that are exchanging data packets. It then verifies if each IP observed after the learning phase belongs to this list, otherwise it generates an alert.
2. *Payload whitelist*: Retrieves from the constructed normal behavior baseline all the combinations of data for each container of the Geonetworking, CAM and DENM packets. It then compares the containers of the normal behavior baseline against the containers

observed in each packet after the learning phase. When they don't match an alert is generated.

3. *Frequency analysis*: Calculates the time between two subsequent packet occurrences of the same IP address. This is done over all the messages in the constructed normal behavior baseline and in the end an average of the observed frequency is computed. The average is then compared against the time interval for each set of two subsequent packet occurrences with the same IP and if the difference is >10% then an alert is generated. This algorithm is effective for periodic transmissions, which is often observed in safety oriented V2X packets [31].

4. *Time inter-arrival analysis*: Defines windows of predefined time interval size (e.g. 1 sec). For each window it calculates an single score for all the packets that belong in that window based on the periodicity of packets as well as the frequency of changes in their payload [66]. When the score in a window is significantly higher than the ones computed during the learning phase windows, then an alert is generated with the anomaly that is spotted.

5. *Payload inspection*: Configured according to the ETSI EN 302 637-2/3 specifiying the exchanged data through the GeoNetworking, CAM and DENM packets to identify if the received packets after the learning phase comply to the format and expected values. If a packet does not comply an alert is generated.

Non-safety oriented V2X packets

Packets that are sent through a normal TCP or UDP connection are also part of the normal behavior baseline that is stored in the EventDB. As the data formatting inside these packets is not following a specification and is application-specific, only behavior-based IDS techniques were applied. Hence only algorithms 1 (*IP whitelist*), 3 (*Frequency analysis*) and 4 (*Time inter-arrival analysis*) that are applied to the safety-oriented packets are used.

2.1.3 Experiments

In this section we present the experiments that were conducted using attacks that starting by obtaining access to the wireless network and sending packets to OBU port (Section 2.1.2) of every vehicle. As the NIDS is integrated in the central gateway it can detect attacks and upon detection the central gateway can choose to drop the packet instead of taking actions in the internal in-vehicle architecture based on its information.

Attack set

The attack set that we used for testing the NIDS was focused on the safety-oriented V2X packets, as the non-safety are based on manufacturer implementations and a generic attack set is challenging to be established. Additionally, the safety-oriented V2X attack set is using the two main V2X connectivity interfaces that were presented in D2.2 [14]:

- Manual GeoNetworking protocol implementation
- The Vanetza open-source implementation of the ETSI C-ITS protocol suite.

Attack name	Protocol	Description	Consequence
DoS	GeoNetworking	Network flooding with empty V2X packets	Dropping of legitimate V2X data
Frame injection	CAM	Setting vehicle speed to zero (vehicle standing still)	Nearby vehicles may engage the breaks
Replay	CAM	Impersonate a moving vehicle	Confusion to nearby vehicles
Frame injection	DENM	Inject false warnings for road accidents	Road traffic/confusion to nearby vehicles

Table 2.1: Considered attacks and their consequences

We have extended the V2X connectivity libraries with scripts for the attacks that are presented in Table 2.1.

An example of a script that launches an attack is shown in the following code fragment. Specifically, the fragment focuses on the configuration of the CAM packet used as a part of the frame injection attack that sets the vehicle speed to zero.

```
1  CoopAwareness_t& cam = message->cam;
2  cam.generationDeltaTime = gen_delta_time * GenerationDeltaTime_oneMilliSec;
3
4  auto position = positioning_.position_fix();
5
6  BasicContainer_t& basic = cam.camParameters.basicContainer;
7  basic.stationType = StationType_passengerCar;
8  basic.referencePosition.altitude.altitudeValue = AltitudeValue_unavailable;
9  basic.referencePosition.longitude = position.longitude.value();
10 basic.referencePosition.latitude = position.latitude.value();
11 .....
12 cam.camParameters.highFrequencyContainer.present =
13 HighFrequencyContainer_PR_basicVehicleContainerHighFrequency;
14 BasicVehicleContainerHighFrequency& bvc =
15 cam.camParameters.highFrequencyContainer.choice.basicVehicleContainerHighFrequency;
16
17 bvc.heading.headingValue = 0;
18 bvc.heading.headingConfidence = HeadingConfidence_equalOrWithinOneDegree;
19
20 //speed set to zero by attacker
21 bvc.speed.speedValue = 0;
```

Listing 2.1: CAM frame injection attack example

As illustrated in *lines 6-10* the attacker identifies itself as a passenger car in order to be included in the V2X system. Afterwards, in the high frequency container (*lines 14-21*) the heading as well as the speed of the vehicle is set to zero. This will indicate to the nearby vehicles that the vehicle is stationary. As a consequence, the nearby vehicles may engage the brakes to avoid a potential crash.

Results

In this section we present an overview of the detection results that we have obtained through the NIDS for the attacks that were presented in Table 2.1. In particular, we focus on the attack that is presented in Listing 2.1. This attack cannot be easily detected by the specification algorithms, since setting the speed to zero is a legitimate action. Hence, we focus below on the

detection of the attack using the time inter-arrival analysis module.

In the left part of Figure 2.4 we illustrate the scores of all windows when being in the learning phase (no attacks are present). We observe that the score over the total number of windows over the entire learning phase has its highest value set to 80. This is defined as the maximum score (maxScore) and is used afterwards by the detection module.

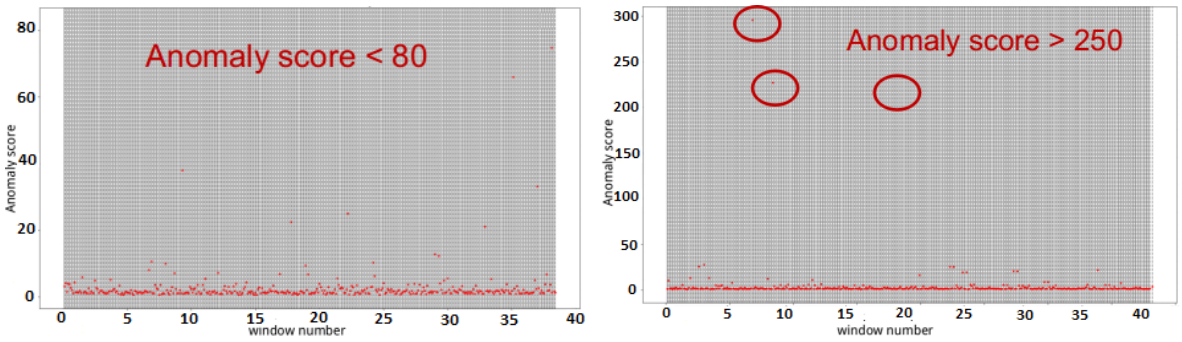


Figure 2.4: Time inter-arrival analysis before and after the attack

In the right figure we can observe a data capture with the presence of attacks after the learning phase has elapsed. These attacks are happening in the [5-10] sec and [15-20] sec interval. They are spotted by the detection algorithm through the presence of a threshold for identifying anomalies that is computed by: threshold = maxScore + margin, where maxScore is defined as above and margin is given by: (1/4)*maxScore to provide a tolerance margin for the algorithm.

Then, the algorithm uses the computed threshold to compare it against the score of every window. Specifically, if the score of a window is greater than the computed threshold, then this window is flagged as having an anomaly. This is observed in the right part of Figure 2.4 as the score is >250 in the [5-10] sec and [15-20] sec windows.

2.1.4 Discussion

The NIDS presented in this section allows to secure V2X connectivity interfaces by detecting attacks before they proceed to the in-vehicle architecture. To accomplish that, it includes modules to compliment the in-vehicle NIDS that was presented in Deliverable D1.3.

Though it is effective in detecting anomalies that are altering the normal behavior, the NIDS may not keep all the contextual information that are relevant for investigating if the anomalies are due to an attack, an operational fault or even a legitimate behavior. This may lead to the gateway dropping packets that are legitimate. Therefore, the central gateway should also be complimented with solutions to first analyze the NIDS alerts and then take respective action. Solutions to address this have already been investigated and hence can be used to compliment the NIDS [59].

Moreover, to ensure a higher level of end-to-end security other mechanisms can also be considered, such as encryption on the exchanged V2X packets through TLS or IPsec protocols or enabling firewall rules in the central gateway to for instance discard V2X packets from blacklisted IPs [49]. In each implementation scenario the central gateway should ensure the security solutions as complementary to each other and do not overlap or tamper with the functionality of the others.

2.2 Anomaly Detection - based Machine Learning

In this section, we provide another perspective to solving vehicular security challenges. In-vehicular networks and largely adapted CAN bus sub-networks are usually the weakest links when it comes to a potential plethora of attacks committed in the connected vehicle domain. Attacks could enter from the ex-vehicular domain in unprecedented proportions as the legitimacy of CAN communication becomes more ambiguous across different vehicle manufacturer models. CAN data is also subject ambiguity per its interpretation across vehicle manufacturers that could tie different meanings to a same type of CAN ID and payload combination. Therefore, we deliver a modern solution that tackles the problem for as many vehicle manufacturer models as possible, all while limiting the black box characteristics of decades-old in-vehicular security.

We aim to accomplish two integral research outcomes. The first outcome is an IDS than can serve as a strategy tool for evolving ML solutions to DL realms. The second outcome is an IDS that can also detect attempts to manipulate the data mining operations and even parts of the ML decision-making. We demonstrate one way to accomplish the first outcome in the following sections, and discuss the second outcome in later sections.

2.2.1 Architecture

We designed a prototype ML framework with prior research outcomes in mind, as shown in Fig. 2.5. Modules are numbered to indicate different routes taken over the course of framework training and testing phases.

Our ML framework is a virtual combination of reproductive and adversarial learning paradigms. By reproduction and adversariality, we refer to "learning representations by competition". We aim to simplify the emulation of human-like doubt (directly observed elements versus a number of pre-combined estimations towards directly observed relationships) in our learning paradigms to encourage the presence of human-in-the-loop scrutiny whenever needed. We exploit inconsistency and consistency of learning by breadth, seeing as depth cannot be an elementary part of our framework unlike in traditional deep learning solutions. Exploring and exploiting CAN, and possibly CAN-Ethernet data, could pose major risks in false interpretation if we aim to have a learning model that does not require tedious runs over and over again.

Programming tools and libraries used to implement and demonstrate the outcome of our solution design are primarily Python-based, such as Jupyter Notebook, TensorFlow, deep learning API Keras [18], machine learning library Scikit-learn [54], and data analysis library pandas [45].

Preprocessing

Our training data is preprocessed according to the level of detail. Data were acquired in open-source text and comma-separated value sheets from CAN bus data pools by Han et al. [29], Lee et al. [39], and Seo et al. [64] from Hacking and Countermeasure Research Lab (HCRL)[1][2][3].

[1] http://ocslab.hksecurity.net/Datasets/survival-ids
[2] http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset
[3] http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset

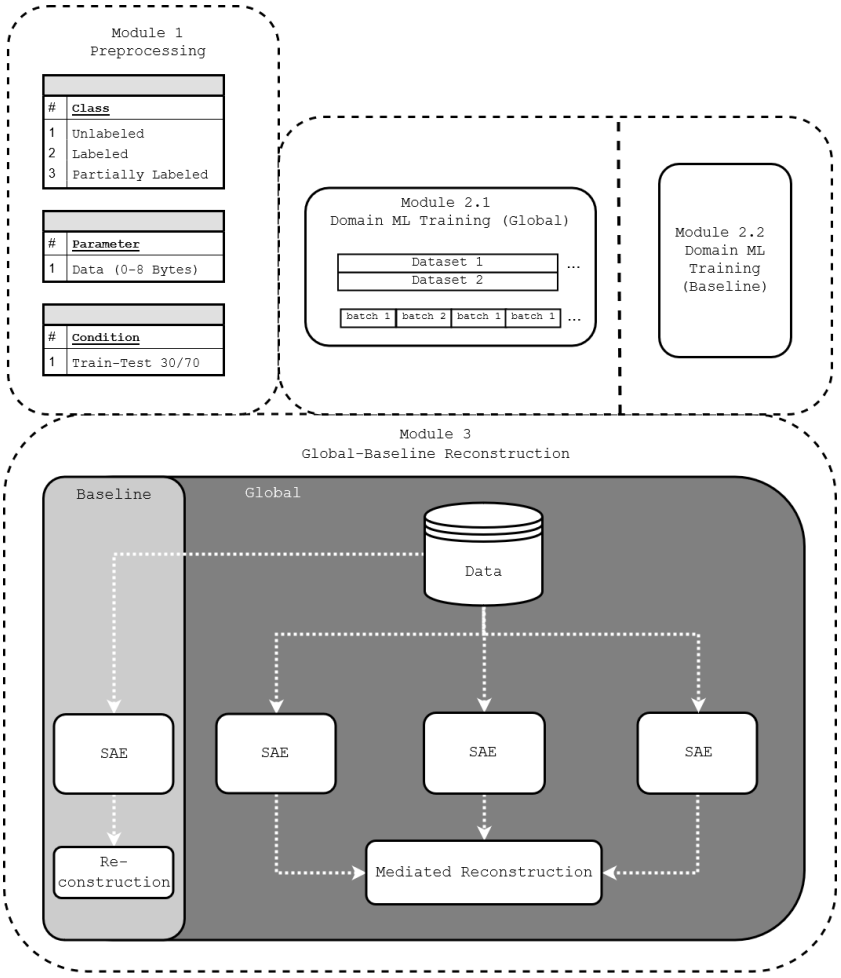


Figure 2.5: The proposed ML framework.

Target vehicles comprise manufacturer models such as KIA Soul, Hyundai Sonata, and Chevrolet Spark . Each log contained a set of parameters often differing from one another but typically containing both ID and data payloads (Fig. 2.6).

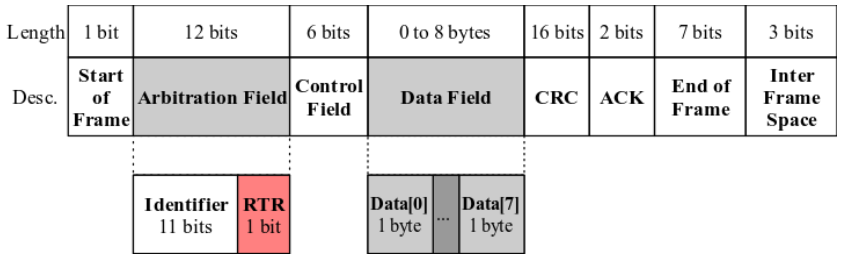


Figure 2.6: CAN bus message layout. [39]

Logs either included or excluded the following types of CAN attacks (Fig. 2.7 and Fig. 2.8):

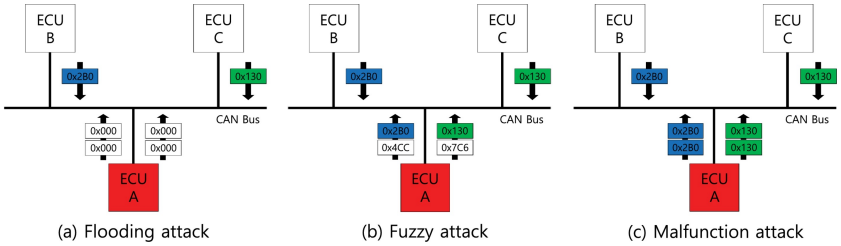


Figure 2.7: Flooding, Fuzzy, and Malfunction attacks on the CAN bus. [29]

- Fuzzy attacks constitute indiscriminately compromising injections of random CAN packets. This attack can be performed by generating random existent and nonexistent CAN ID and payload bytes for every 0.0003 seconds.
- DoS (Denial of Service) attacks manipulate the message arbitration process to maintain dominant status on the CAN bus for an unspecified amount of time. Communications can be restricted between the ECU nodes to compromise the normal driving routine. This attack can be performed by flooding a large number of messages with the lowest CAN ID value possible (0x000) into the vehicle networks. The lower ID value constitutes a higher priority for CAN message arbitration.
- Malfunction attacks hijack select CAN IDs to compromise select message sequences. To perform this attack, the attacker should manipulate the data field at the same time.
- Impersonation attackers inject illegitimate messages under the guise of another legitimate node identity.

We collected approximately one second worth of CAN messages (2000-2500 messages), which is more than enough to derive some useful points from an analytical perspective:

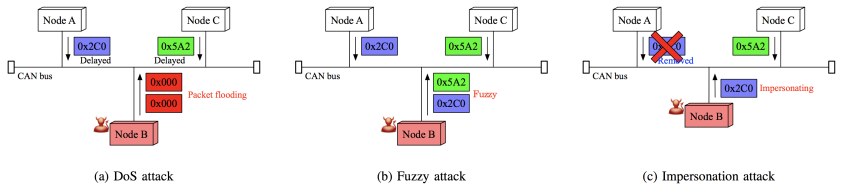


Figure 2.8: DoS (Flooding), Fuzzy, and Impersonation Attacks [39]

1. CAN IDs have different meanings across different vehicles. According to discoveries by Miller and Valasek [46], the ambiguity of meanings would trivialize the successful execution of desired actions after accessing the CAN bus. Thus, both illegitimate and legitimate entities can end up in the same bind.

2. Traditional outlier detection over uncompromised data with all parameters is not enough if we aim to solve problems with a multitude of attacks. Based on prior findings by Markovitz and Wool [42] and our observations, latent behaviors in CAN payloads (cycles and arrhythmic patterns) can be leveraged to expand the learning surface of our research problem.

3. Parts of our open source CAN data were classified in advance. However, this is still not enough to train a trustworthy IDS.

Domain Training

In the second module, we provide an approach appropriate for our research strategy. Our experiments, surveys, and brief interviews were critical in motivating how we solve our research outcomes.

Logic and execution similar to our chosen approach can be discovered in efforts such as online network intrusion by Mirsky et al. [47], ensemble clustering by Liu and Lam [40], and co-clustering for multi-task learning by Murugesan et al. [48]. We also observe gradual trends from Google [55] and Uber [25] in ML towards semi-supervision and federated machine learning. In semi-supervised learning, we emulate real-life thinking by generalizing between a small amount of classified data and large volumes of unclassified data to achieve new insight.

We identify the following steps to achieve the chosen approach. Firstly, we separate our data pool to have less training data than testing data. We can accurately observe the estimation capabilities of our approach thereafter. Once the data is separated, we establish separate domains to solve both multi-class and one-class problems in the same scope of research. In this respect, we moved data for direct learning purposes to **baseline domain**, and data for nonlinear learning to **global domain**, respectively. With these names, we refer to easily identifiable and vaguely distinguishable patterns of CAN data and ID fields. More specifically, we designate the following tasks:

- In the global domain, the learning process by generating data fields astochastically. Unlabeled, mislabeled, and labeled instances are handled by adopting Autoencoders (AE)

[33] (Fig. 2.9), and particularly their sparser varieties (SAE) [2] (Fig. 2.10), through which Keras-generated data is "imperfectly" reproduced [10].

- In the baseline domain, the learning and decision processes are concurrently performed on the outcome with respect to linearly distinct traits of the data. Linear regression or transformation can be implemented to assess the mutual ID-payload behavior.

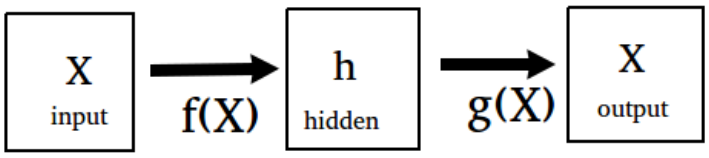


Figure 2.9: A diagram example of the typical AE. AEs assume the *input - hidden - input* transformation, which would benefit learning important properties of data. The properties to be learned in turn depend on how strict is the AE operation in overall. [2][68][15]

Our research entails an idea that reproducing events can be safer as a judgment tactic than learning their associations, due to liberties in restricting and derestricting the pool of definitions. One analogous example is that how humans manage to learn effectively via the selection bias. Our preprocessing procedure entertains the similar idea of biases as follows:

1. A small number (2) of pseudolabels was injected into most of the unlabeled datasets. This results in less trustworthy, albeit largely directional estimates. By directionality, we entertain at least one possibility where some data inputs might be falsely labeled to manipulate other framework evaluations. This is similar to how humans would naturally suspect anything in their assumingly safe and secure surroundings.

2. The remaining unlabeled datasets were partially labeled according to descriptions of typical CAN attack manifestations [39]. This is similar to how we attribute previously learned dangers to new environments where certain events may manifest in a similar fashion.

3. Labeled datasets should be preferably devoid of anomalous messages. This is similar to how humans may recognize obvious deviations from their prior perceptions as a potential threat.

Global-Baseline Testing

In the last module, we outline the means to process the outputs of SAEs.

As previously mentioned, we steer the focus from ground truths to reproduction and its quality. Hence, we can evaluate this quality only through differences, or how far from the regression line our data points are. Distances can be depicted either as a mean square (MSE), a root

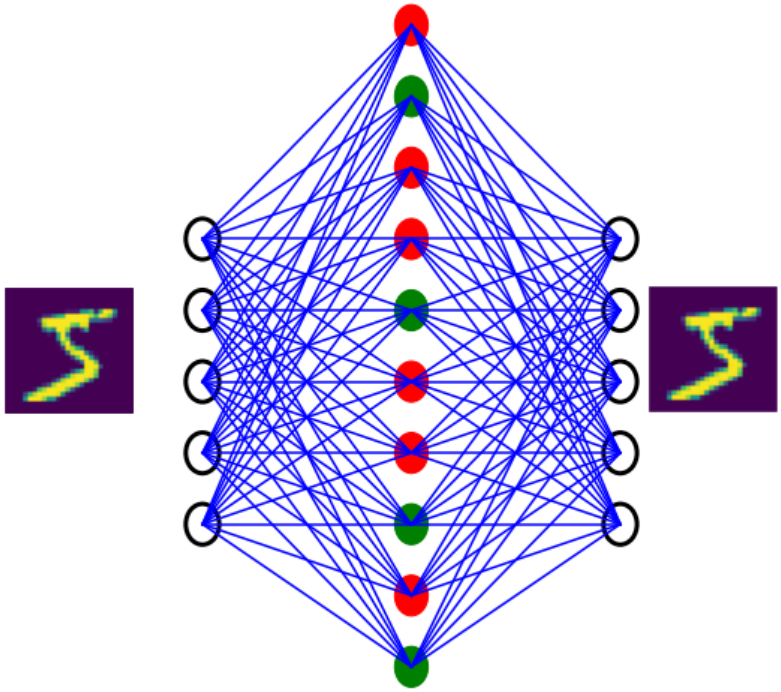


Figure 2.10: An example of the typical SAE. The middle layer represents the hidden layer, whereas the green and red nodes constitute the deactivated and activated nodes respectively. Unlike AE, SAE incorporates a single hidden layer with more variations in activated units than the input layer. While not in-depth, optimally trained variations in sparse representations can help us to extract the meaning that can be influenced directly. [2]

mean square (RMSE), or a mean absolute (MAE) of their deviations [69]:

Mean squared error	MSE	=	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	RMSE	=	$\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	MAE	=	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$

- MSE is widely used due to its computing speed and easier manipulation than other measures. However, due to a lack of scaling to the original error, MSE values cannot relate to the original scale.
- MAE is usually decent for accuracy measurements. However, if we had an MAE of 20 for a particular output data instance, we would have no means to determine whether they are facing a desirable effect or a failed estimate. If we reproduced over 2000 data instances as we did in our experimentation scenario, it could raise some concerns. With lower volumes, however, it just spells poor model performance.
- RMSE is simply a root of MSE that allows relatively high weights to large errors and is useful when large errors are undesirable. Both MAE and RMSE can be adopted together to diagnose variations in the forecast errors. RMSE is always larger or equal than the former, and the greater the difference between these measures, the greater the variance can be determined in the individual errors of the sample. All the errors share the same magnitude if they are equal otherwise.

According to [17][70], neither one of the measures constitutes a one-size-fits-all indicator.

2.2.2 Results

In this section, we decide how and against what exactly are we testing our established framework architecture.

First, we pose a temporary question about the inevitability of the baseline domain. Considering that a delay is almost always warranted for reproductions in real time, we determine a number of global SAE combinations that can be varied according to their timely and effective performance:

- L: a global SAE trained on labeled data.
- U: a global SAE trained on unlabeled data.
- P: a global SAE trained on partially labeled data.

- LUP: global SAEs trained on labeled, unlabeled, and partially labeled data.
- LU: global SAEs trained on labeled and unlabeled data.
- LP: global SAEs trained on labeled and partially labeled data.
- UP: global SAEs trained on unlabeled and partially labeled data.

Metrics used to evaluate outcomes against one another are Area under the Curve (AUC) and Average Precision Scores over time series. AUC represents the degree of separability, or how much our framework is capable of distinguishing between labels. AUC is also often considered as a part of Receiver Operation Characteristic (ROC) curve measurements, both of which form a probabilistic type of discriminant relationship. Discriminating capability precedes the presence and successful retrieval of relevant information by the IDS, also known as Precision and Recall, respectively [16].

We consider three possible scenarios our comparative evaluations. We can either keep the baseline reproduction (Fig. 2.11 and Fig. 2.12), exclude it (Fig. 2.13 and Fig. 2.14), or use one of the randomly chosen global SAEs as the baseline (Fig. 2.15 and Fig. 2.16).

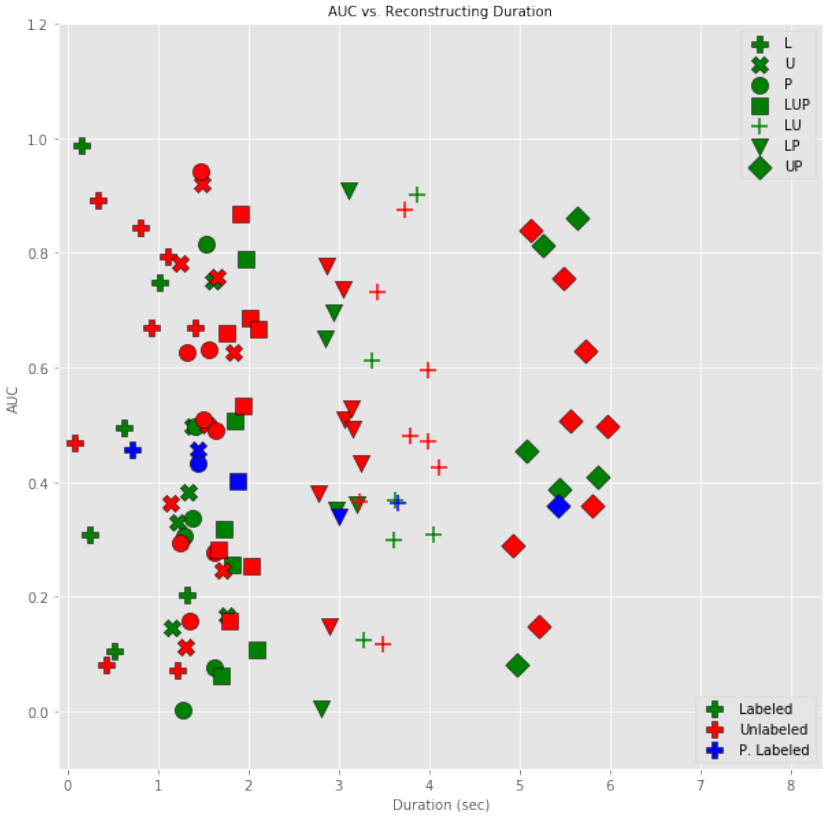


Figure 2.11: AUC values against time elapsed for reproducing new data over both global and baseline domains.

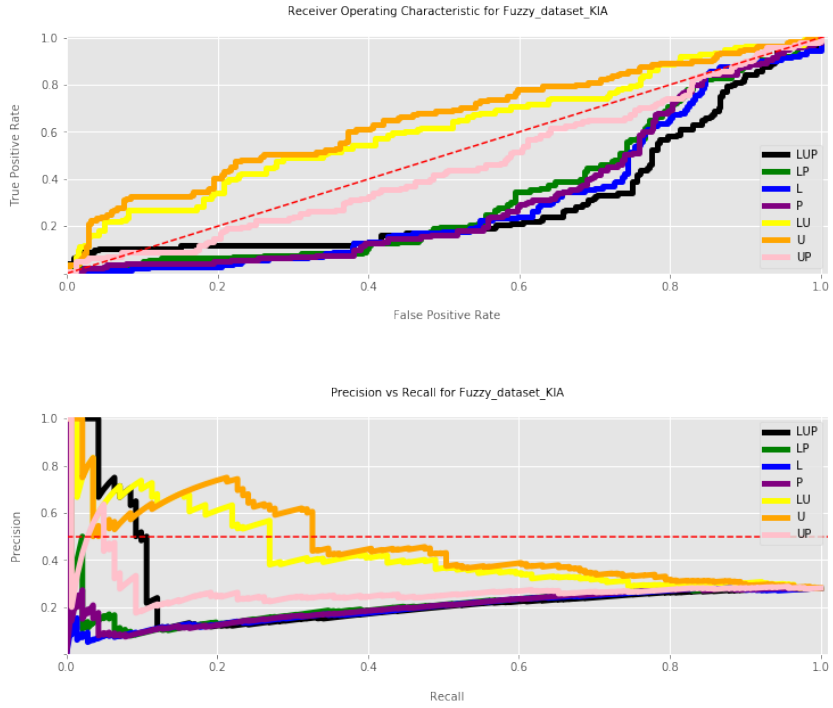


Figure 2.12: ROC and Precision-Recall curves over both global and baseline domains.

We cab observe possibilities to attain higher AUC rates with lower rates in testing time by opting one "branch" of global SAEs rather than all concurrently. From the certainty perspective, however, obvious clusters nearing the AUC values of 0.4-0.6 and down curves in both ROC and Precision-Recall planes indicate that the learning model of our framework may require more iterations in reproduction and more elaborate, turn-based strategy between SAEs and their combinations.

In case we depend on the original input values instead, sufficient accuracy could be achieved more efficiently. However, accuracy in detecting legitimate anomalies can become equivocal when compared to detecting assumed and false anomalies.

While ROC curves and Precision-Recall curves indicate that any select combination of global SAEs could lead to the said level of accuracy, another domain must established to prevent unreliable guesses on observed events from the engineering side.

The last and the least orthodox option would be selecting one of the best global SAE candidates as the baseline. Prior figures depict how distinct clusters appear on the upper left side of accuracy-efficiency plane, and at least 85% of the global SAE combinations are shown to be way ahead of the most acceptable thresholds in ROC and Precision-Recall estimations. Engineers may still have to design a fine-tunable RNG module or technique in advance to leverage the 85% batch of global SAEs improving the learning performance, and simultaneously prevent the rest of 15% global SAEs from inducing more redundancy in reproductions.

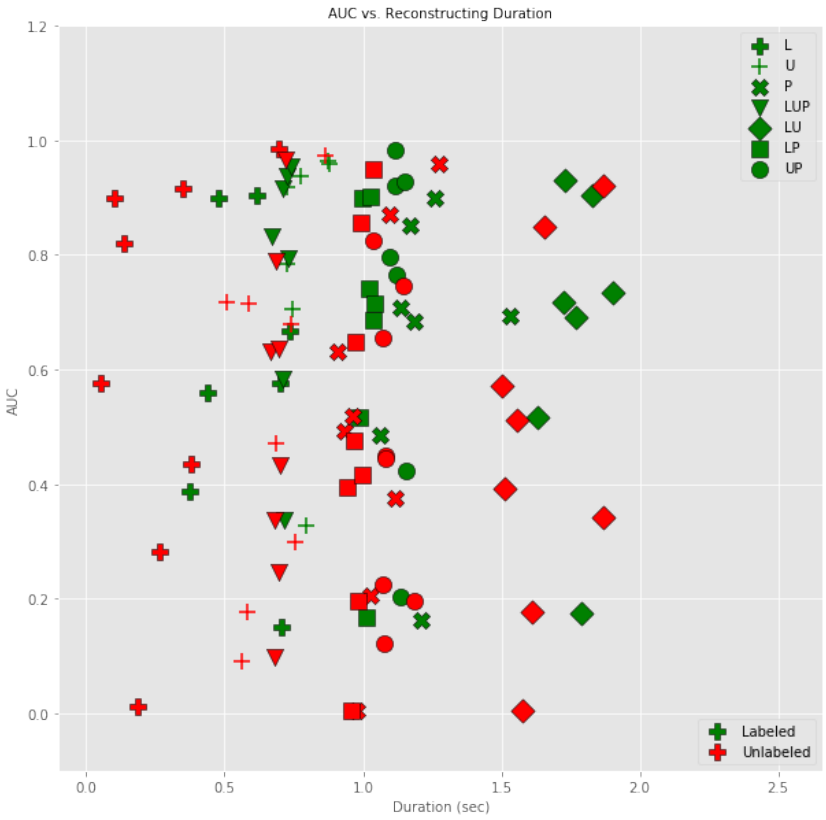


Figure 2.13: AUC values against time elapsed for reproducing new data over global domain.

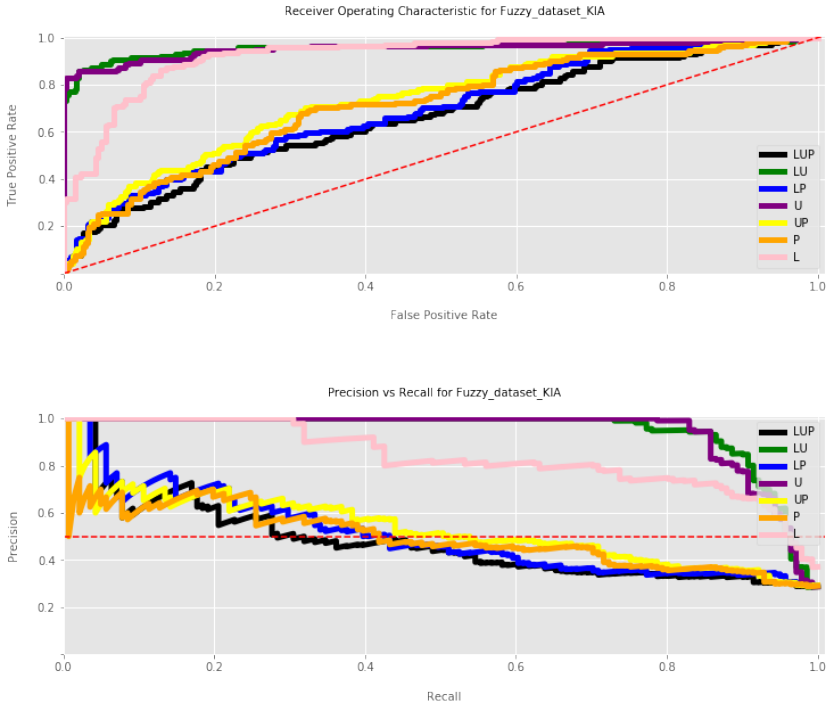


Figure 2.14: ROC and Precision-Recall curves over global domain.

2.2.3 Discussion

In this concluding section, we revisit our research outcomes.

Liberties in deciding acceptable observations should be strongly considered for synchronizing accessibility and humanly coherent decision-making with machine learning. The future of IoT and its vehicular umbrella (IoV) will be home to a plethora of cutting-edge and quickly resolving ML solutions. We are still unable to determine whether we can jump in anytime we want or need to. We are also unable to reason with not intervening at all. Our framework is not the final answer to these doubts with human analogies and easy-to-comprehend ML (such as SAEs). We consider our prototype research the first success in delivering a potentially effective tool to bridge the base knowledge of white-box ML systems with more advanced black-box DL solutions.

Further research has to be conducted in order to warrant real-time performance. This may not be always necessary, though. For instance, on-off type of adoption could prove to be possible and likely useful at the moment. We can plug in whenever there is a need to cross-examine CAN data and either delegate further actions between vehicle-based IDSs and Intrusion Prevention Systems (IPS).

Trustworthiness is one crucial topic that requires critical assessment in the same umbrella of problems explored and exploited in our ML framework. Adversaries are more likely to hijack the core operations of our IDS rather than the vehicle it is supposed to protect in the future. It is up to the course of current research endeavors to determine how we can build an intelligent line

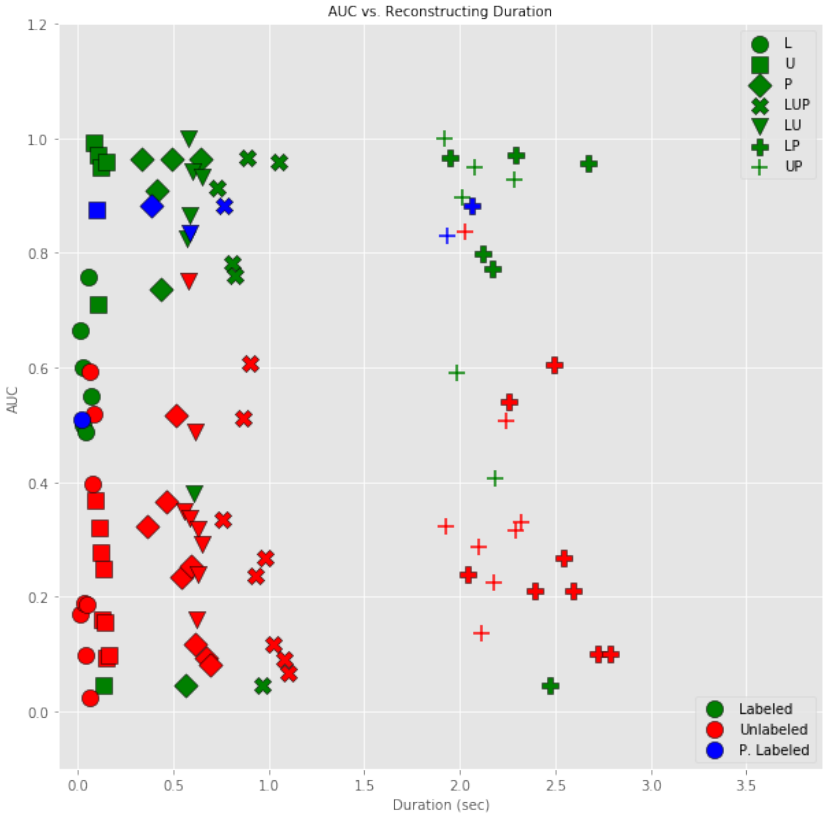


Figure 2.15: AUC values against time elapsed for reproducing new data over global domain and a single, randomly chosen global SAE.

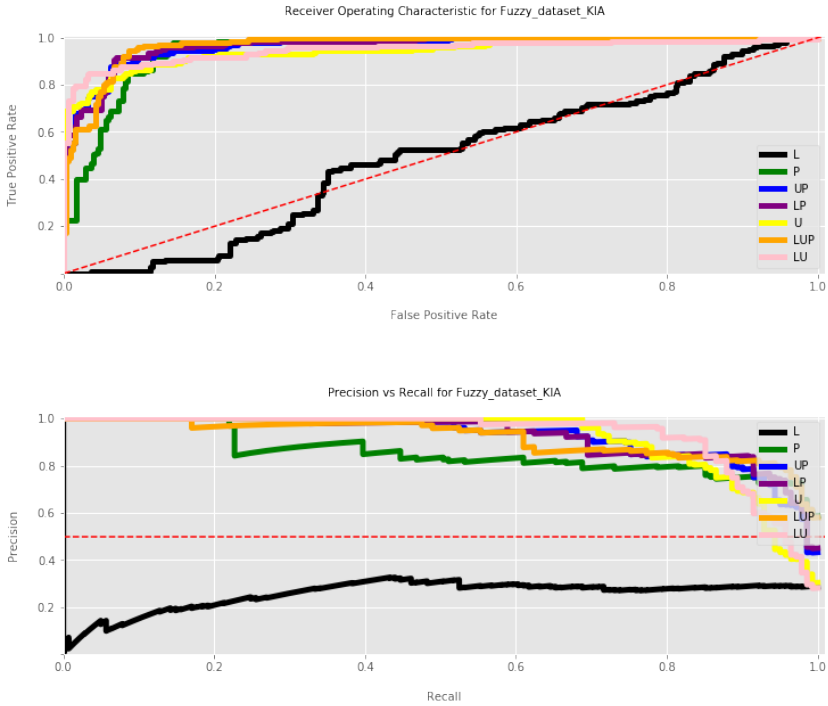


Figure 2.16: ROC and Precision-Recall curves over global domain and a single, randomly chosen global SAE.

of defense in the connected vehicle ecosystem. For example, one may dynamically separate and migrate statistical and framework learning organs to ensure secure and up-to-date operations.

3 Authorization Framework for Cooperative Intelligent Transport Systems

3.1 Introduction

Intelligent Transport Systems (ITS) are "*systems in which information and communication technologies are applied in the field of road transport, including infrastructure, vehicles and users, in traffic management and mobility management, as well as for interfaces with other modes of transport*" [5]. Cooperative Intelligent Transport Systems (C-ITS) aims to improve the quality of ITS through the use of sensing capabilities and advanced information and communication technologies [26].

Significant developments have taken place over the past few years in the C-ITS domain. Several initiatives and projects have been established all over the world (e.g., DITCM [61], CONVERGE [1], US-ITS [3]) to enable the development of a cooperative architecture to support the communication of vehicles with the transport infrastructure, service providers and other vehicles. While these initiatives provide a foundation for the design and development of C-ITS, their results can be deployed at a large scale only if the developed infrastructure and services meet the requirements posed by the C-ITS domain, including scalability, performance and security.

Security is particularly challenging to achieve within C-ITS as it encompasses multiple systems, such as automotive systems, road infrastructure, services and applications, and requires addressing attackers with various motivations and levels of skills, and diversity of threats and countermeasures [38]. Among the several security concerns, the protection of information gathered and shared within C-ITS is of utmost importance to enable its deployment at a large scale. Typically, sensitive data are protected through the adoption of authorization mechanisms that guarantee that only authorized parties can gain access to the data. While several authorization frameworks have been proposed to address authorization concerns in several application domains, there has been very little attention towards authorization in the C-ITS domain. Given the critical and dynamic nature of C-ITS, authorization mechanisms should not affect the functioning and performance of the system as well as provide fine-grained protection of sensitive information and resources. Specifically, an authorization framework for C-ITS should allow the specification and evaluation of context-aware policies to deal with the dynamicity of C-ITS, minimizing the overhead of the authorization process, and guaranteeing its reliability [56].

We present an authorization framework that addresses the unique challenges of the C-ITS domain. The design of our framework leverages principles of both policy-based [50] and token-based [4, 21] architectures to deal with the dynamicity of C-ITS while minimizing the overhead introduced by the authorization process. Specifically, we decouple the evaluation of policies from their enforcement. Our solution encompasses a policy-based authorization server that is used off-line to generate tokens encoding user permissions based on the policies provided by the resource owner. Tokens are then locally validated by the resource server at request time to determine whether access should be granted. While this decoupling allows minimizing the

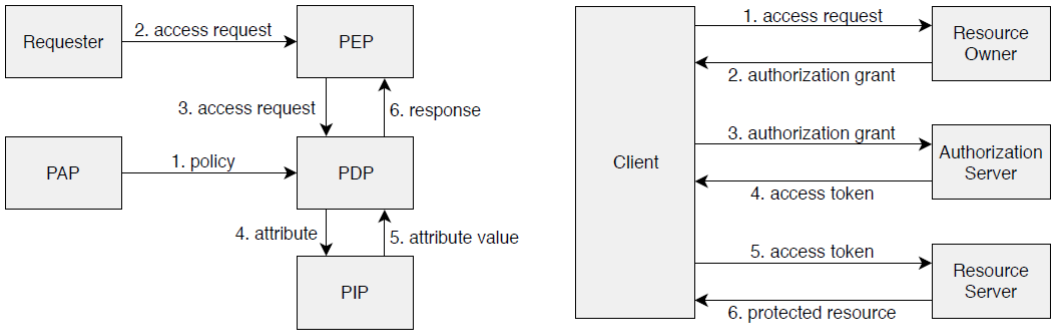
overhead introduced by the authorization process at request time, relying only an off-line policy evaluation does not make it possible to account for access constraints based on the run-time environment. To this end, we devise authorization tokens that encompass constraints to be verified at request time. For the design of our authorization framework, we leverage a C-ITS reference architecture as a baseline. The adoption of a C-ITS reference architecture helps identifying the C-ITS systems involved in the authorization process and, thus, facilitate the realization and integration of our authorization framework within existing C-ITS deployment sites.

3.2 Related Work

Authorization services (the focus of this work) aim at the protection of sensitive information exchanged within C-ITS (e.g., location data). A typical solution to protect sensitive information and resources is through the adoption of access control solutions that guarantee that only authorized parties can gain access. Access is regulated using policies that specify which actions an entity can perform on a certain object. In the remainder of the section, we provide an overview of the reference architectures commonly adopted for the design of authorization frameworks and discuss the main challenges to be addressed in the design of an authorization framework tailored to C-ITS.

Authorization Reference Architectures: Several architectures have been proposed for the design of authorization mechanisms. Two widely adopted architectures are the *policy-based* and *token-based* architectures. Policy-based architectures can be exemplified by the reference architecture proposed by XACML [50], the de facto standard for the specification and enforcement of attribute-based access control policies. This architecture comprises four main components: *Policy Enforcement Point* (PEP), which provides an interface with the system and is responsible for enforcing access decisions; *Policy Decision Point* (PDP), which evaluates access requests against access control policies and determines whether access should be granted or denied; *Policy Administration Point* (PAP), which acts as a policy repository and offers facilities for policy management; *Policy Information Point* (PIP), which denotes the source of information (e.g., context information) needed for policy evaluation. Figure 3.1 shows the interaction between these components. The PAP makes the policies available to the PDP (1). Upon receiving an access request (2), the PEP forwards the request to the PDP (3), which evaluates the request against the policies fetched from the PAP. If additional information is required for policy evaluation, the PDP queries the PIP (4,5). The PDP evaluates the request against the policies and returns a response specifying the access decision to the PEP (6), which enforces the decision.

Authorization mechanisms adopting a policy-based architecture typically provide a single, centralized point for the evaluation and enforcement of access control policies [34]. This solution may not be suitable when resources are distributed across different nodes, which is a typical situation in C-ITS. The last years have seen the emergence of token-based architectures as an alternative to policy-based architectures to deal with the needs of open and decentralized systems. Various standards have defined reference token-based architectures and authorization protocols [4, 21]. Although these architectures and protocols vary in the way tokens are generated and in the flow of the authorization process, they share the same underlying principles. As an example, Figure 3.1 presents the OAuth protocol [21], in which a client application first obtains a token encoding its permissions from an authorization server and subsequently uses it to access a given resource.



(a) Policy-based Architecture (b) Token-based Architecture

Figure 3.1: Authorization Reference Architectures

Challenges: An authorization framework should not affect the functioning of C-ITS. Given the constraints imposed by these systems, the design of an authorization framework for C-ITS presents a number of challenges. These challenges will be used to identify the main concepts and design principles that should be considered when developing an authorization framework for C-ITS.

- *Dynamicity:* C-ITS are complex and dynamic systems in which an increasing number of entities (e.g., vehicles, RSUs) are connected and in which network topology and connectivity changes over time. To handle the dynamic nature of C-ITS, an authorization framework should support the specification and evaluation of context-aware access control policies that impose conditions on the C-ITS ecosystem such as access time and location.

- *Management:* The dynamicity of the C-ITS ecosystem can also affect policy management. In such systems, the resources of an entities can be stored and managed by different administrative domains interacting together. Therefore, an authorization framework should be able to support the management of access control policies for devices and resources across multiple domains.

- *Automation:* A main characteristic of C-ITS is collaboration, which is achieved through interactions between entities involved in the C-ITS (e.g., vehicles, RSUs). These interactions involve the sharing of real-time safety critical information. Hence, C-ITS systems require a high level of automation, possibly without any user involvement. This need for automation also reflects in the authorization process.

- *Performance:* C-ITS are critical systems in which delays can have serious consequences and even result in human loss. Therefore, services deployed within the C-ITS should not introduce latency both in terms of computation and communication. This constraint extends to the authorization process. In particular, the authorization process should not inhibit performance with significant overhead, which violates the timing constraints imposed by the C-ITS.

- *Reliability:* The critical nature of C-ITS also poses high demands for business continuity, even in cases of system failures. On the other hand, the highly sensitive information

gathered and exchanged within the C-ITS requires protection and its disclosure to unauthorized parties should be prevented. Meeting these (apparently conflicting) demands requires the authorization process to be reliable. Even in cases where failures or loss of connectivity occur, the authorization framework must still be operational.

Several security services for ITS have been proposed in the literature [7, 38, 56, 57, 65, 67]. However, they typically focus either on authentication alone or on the protection of communication using cryptographic techniques. To the best of our knowledge, only a few authorization frameworks have been proposed in the ITS context. Salonikias et al. [60] propose a policy-based authorization mechanism tailored to vehicular infrastructures based on fog computing. This mechanism comprises multiple PDPs and PEPs located at the edge, while a single PAP (which also encompasses a PIP) deployed in the cloud is responsible to maintain and propagate access control policies to the PDPs. Gupta et al. [28] present an authorization framework for Internet of Vehicles. This framework proposes the deployment of authorization components (PEP, PDP, PAP, PIP) at different layers – object, virtual object and cloud level layer – to deal with different types of interactions. Dorri et al. [22] propose an authorization framework for vehicular networks based on blockchain. In this framework, interactions between vehicles are stored in the blockchain as transactions, which are verified by powerful nodes acting as miners. Albouq et al. [9] propose a policy-based framework for ITS infrastructures based on fog computing. Service providers deploy their services in fog nodes and vehicles can connect to these nodes through RSUs acting as edge network units. RSUs rely on the publish-subscribe paradigm to enable vehicles to subscribe to the services deployed in fog nodes. In this respect, the authorization framework resides within RSUs to control which services can be published whereas vehicles can subscribe to any (allowed) service. Riabi et al. [58] propose the use of a distributed hash table (DHT) to handle authorization within ITS. Resources are stored in fog nodes and each fog node maintains a DHT specifying the mapping between fog nodes and the Access Control List (ACL) maintained by them. Upon receiving an access request for a given resources, fog nodes use the DHT to identify the node handling the requested resource and forward the request to such a node, which makes an access decision by evaluating the request against its ACL.

Discussion: Despite the number of authorization mechanisms for ITS proposed by both industry and academia, existing authorization mechanisms are inadequate to deal with the open and dynamic nature of C-ITS systems. Table 1 presents an analysis of existing authorization frameworks for ITS with respect to the challenges discussed in Section 3.2.

An authorization framework for C-ITS should cope with the dynamic nature of ITS. While some frameworks (e.g., [28, 60]) support the definition of context constraints in policies and their evaluation, many frameworks (e.g., [9, 22, 58]) do not, thereby not addressing this challenge. Nonetheless, most frameworks [9, 28, 60] provide a single point for policy administration, thus facilitating policy administration. An exception is the frameworks in [22], in which policies reside within vehicles. It is worth noting that the framework in [58] allows resource owners to deploy their policies to a single fog node and uses a DHT to identify which nodes should evaluate a request for a given resource. However, the DHT stored in each node has to be updated whenever an ACL is modified. The automation of the authorization process is satisfied by all frameworks as they do not require user involvement in the authorization process.

To be effective in C-ITS, an authorization framework should not introduce significant overhead and latency and, in general, should not affect the overall performance of the C-ITS [56]. None of the existing frameworks fully satisfies this requirement. Existing authorization frame-

works typically perform policy evaluation upon receiving an access request, thus delaying service provision. In addition, some frameworks require additional communication to retrieve the context information needed for policy evaluation [60], or rely on technology that is computational expensive like blockchain [22]. Other frameworks [9, 28, 58] adopt a centralized architecture where all authorization components reside within the cloud, a fog node or the vehicle. However, assuming that all (context) information needed for policy evaluation is available from a single source limits the constraints on the context that can be verified.

	Ref. Arch.	Dynamicity	Management	Automation	Performance	Reliability
Salonikias et al. [33]	policy-based	●	●	●	◑	◑
Gupta et al. [15]	policy-based	◑	●	●	◑	◑
Dorri et al. [12]	blockchain	○	○	●	○	◑
Albouq et al. [8]	policy-based	◑	●	●	◑	◑
Riabi et al. [32]	policy-based	○	◑	●	◑	○

Table 1: Analysis of existing authorization frameworks for ITS. Symbol ● denotes that a challenge is *addressed*, ◑ that it is *partially addressed*, and ○ that its is *not addressed*

Existing authorization frameworks partially address the reliability of the authorization process by placing authorization components within the cloud [28, 60], which typically provides recovery measures to ensure business continuity. In addition, Salonikias and colleagues envision redundancy for those components deployed at the edge. Similarly, the frameworks in [22] and in [9] can theoretically ensure the reliability of authorization components by replicating them in blockchain nodes and RSUs, respectively. However, for both frameworks, scenarios of node failure or loss of connectivity are not analyzed. In [58], the request can be sent to any fog node, which forwards it to the node that has the requested resource. However, ACLs are not replicated among nodes, leading to reliability issues in case of connectivity loss or node failure.

In summary, existing authorization frameworks fail to fully address all challenges posed by C-ITS. A main drawback is given by latency due to the choice of a policy-based architecture for their design. In this work, we present an authorization framework for C-ITS that adopt principles underlying the token-based architecture as a baseline for its design (Section 3.4). This architecture provides a foundation to deal with the dynamicity and performance constraints typical of C-ITS scenarios.

3.3 C-ITS Reference Architecture

For the design of our authorization framework for C-ITS and its realization and integration in existing C-ITS sites, we adopt a C-ITS reference architecture as a baseline for our design. A reference architecture is typically used to facilitate communication and cooperation between different stakeholders during the design and development of complex systems. A reference architecture for the C-ITS domain addresses not only demands in the software/system engineering field, but also in traffic engineering, civil engineering, information technology, etc. Moreover, its design should account not only for new systems but also taking into account the infrastructure and systems already in place.

In the recent years, several C-ITS reference architectures have been proposed to address the interdisciplinary concerns and to enable the large scale deployment of region or nation wide C-ITS services. In this work, we adopt the C-ITS reference architecture proposed in the C-MobILE project (`http://c-mobile-project.eu`) as a baseline for the design of our autho-

rization framework. The C-MobILE reference architecture provides a baseline for the design of a C-ITS infrastructure mainly targeting traffic related concerns [19, 36]. The C-MobILE reference architecture is based on the generalization of existing C-ITS architectures while addressing the main concerns of the C-ITS stakeholders.

The C-MobILE reference architecture categorizes C-ITS systems into five main types based on the functionalities they provide, as illustrated in Figure 3.2. Below we present a brief description of the main systems and refer to [19] for details:

- *Support system* consists of systems supporting the governance and management of C-ITS services. Support systems influence all other systems of the C-ITS.
- *Central system* comprises systems that support connected vehicles and roadside units by capturing data from vehicles and roadside units, and providing such data to C-ITS applications. Central systems can be aggregated together or can be geographically or functionally distributed.
- *Roadside system* consists of systems forming the physical road infrastructure such as roadside units, traffic light controllers, and cameras.
- *Vehicle system* comprises systems integrated within vehicles such as a Vehicle On-Board Unit (V-OBU).
- *Traveler/VRU system* consists of personal devices, typically a smart phone or personal navigation device used by a traveler or Vulnerable Road User (VRU).

Security services are provided by the support system. Below we describe its sub-systems.

- *Governance* comprises systems and entities that are responsible for the functioning and security of the C-ITS.
- *Operational Management* comprises systems enabling operational processes such as fault, performance and configuration management of C-ITS systems.
- *Test and Certification Management* supports the registration and management of tested and certified communication systems for ITS (safety) applications.
- *Security and Credentials Management* provides a high-level representation of the systems that enable trusted communications between mobile devices, roadside devices and centres, and protect data from unauthorized access. A sub-systems is the *Authorization Authority*, which is in charge of issuing authorization tickets to ITS entities.

In the next section, we present the design of our authorization framework and show how its components are mapped to the systems of the C-ITS reference architecture. This mapping will help understand the external interfaces, high level functional capabilities of the authorization components within the C-ITS architecture.

3.4 Authorization Framework

Existing authorization frameworks are usually based on either a policy-based or a token-based architecture. As discussed in Section 3.2, policy-based frameworks often introduce delays that

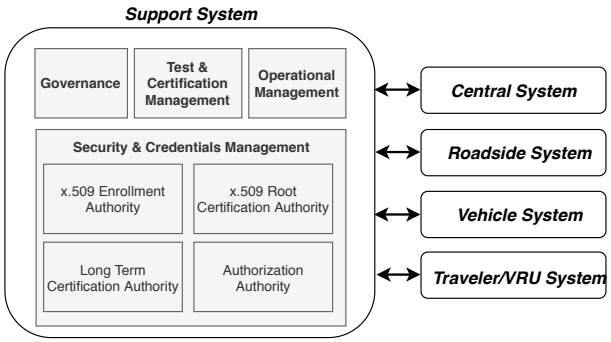


Figure 3.2: C-ITS Reference Architecture

cannot be tolerated by the C-ITS. While existing token-based frameworks address this issue by limiting the operations to be performed at run-time to token validation, they usually require user involvement to determine whether access should be granted, thus providing no automation of the authorization process. In this work, we propose a *hybrid* authorization framework that leverages the advantages of both these architectures. In particular, we divide the authorization process into two main stages: an *off-line* process in which tokens are automatically generated based on policies (without any user involvement) and a *run-time* process in which tokens are validated. Such an approach provides the flexibility and performance necessary to deal with the dynamic and critical nature of C-ITS while providing a high degree of automation. In the remainder of the section, we present the main components of the framework along with the authorization process.

Authorization Components: The authorization framework encompasses the following entities and components:

- *Resource Owner* is the user or legal entity that controls a given resource.
- *Resource Server* is the component hosting the resource on behalf of resource owner.
- *Authorization Server* is the component that protects resources hosted on a resource server on behalf of the resource owner. The authorization server generates tokens based on access requirements specified by the resource owner, thus acting as the PDP.
- *Policy Information Point* denotes the source of context information.
- *Requesting Party* is a user or a legal entity that uses a client application to access resources.

Figure 3.3 shows these components along with their interactions. It also provides their mapping to the systems of the C-ITS reference architecture in Figure 3.2. This mapping identifies which C-ITS systems can play a role in the authorization process.

Authorization process: The authorization process supported by our hybrid authorization framework is performed in two stages. First, the authorization server evaluates the policies off-line and generates an authorization token asserting the permissions of the requesting party. Then, when requesting access to a resource, the requesting party provides the token along with request to the resource server, which validates the token and verifies additional constraints on

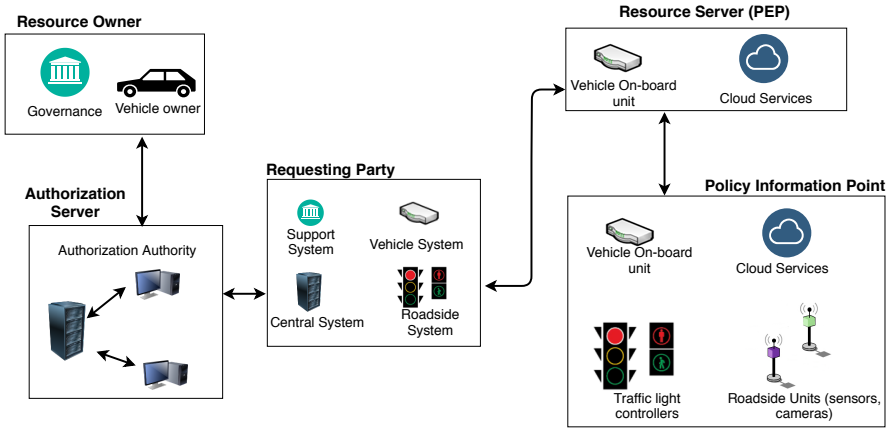


Figure 3.3: Authorization framework and mapping of its components to C-ITS systems

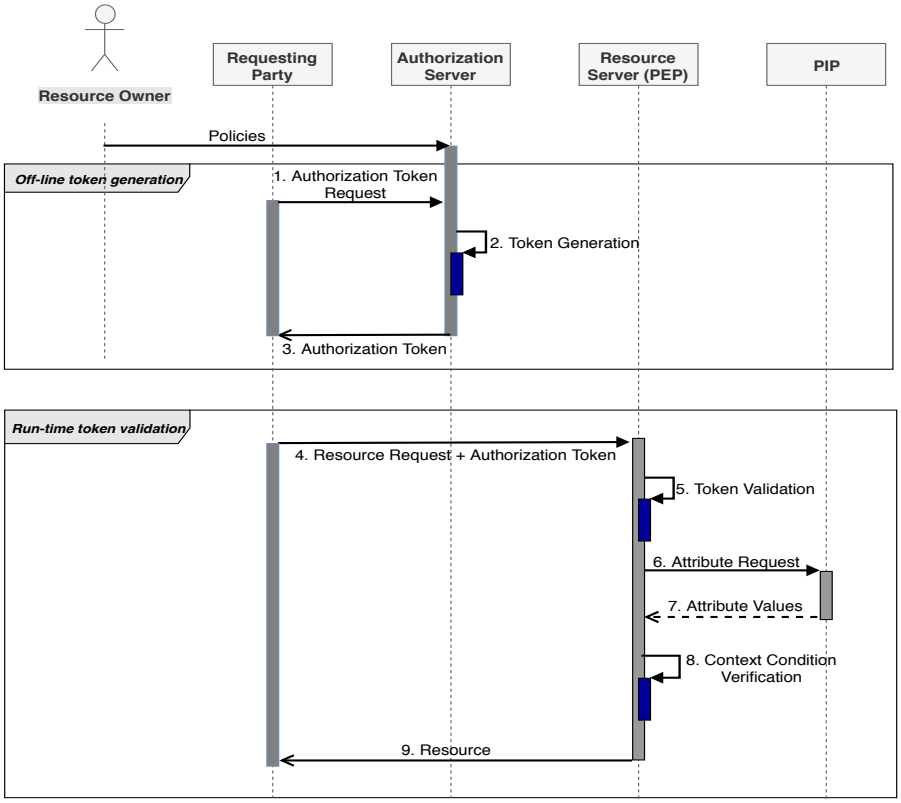


Figure 3.4: Authorization process

the context (if any). The procedures for off-line token generation and run-time token validation are represented in Figure 3.4.

We assume that the resource owner stores her resources in a resource server. Moreover, she has provided the authorization server with access control policies defining who can access her resources. It is worth noting tha resources can be under the control of multiple entities or negotiation between entities may be necessary to determine how resources can be used and

with whom they can be shared. In this settings, data sharing agreements [35, 44] should be established between the involved parties to determine provisions concerning access and dissemination. How data sharing agreements and collaborative policies [20, 41] can be defined is out of the scope of this work and here we simply assume that the policies to be enforced are provided to the authorization server. Interested readers can refer to [52] for a thorough discussion on this issue.

In this work, we consider policies specified in attribute-based access control (ABAC) as this paradigm provides a means for the specification of fine-grained access control policies. In ABAC, access requests and policies are defined in terms of attribute name-value pairs. Policies have a target, which defines the applicability of the policy by specifying to which requests the policy applies, and an effect, which specifies whether the subject has the permission to perform the specified action on the resource (*permit*) or not (*deny*). Figure 3.5 shows an example policy expressed in (a compact representation of) the XACML policy language [50]. This example policy is used to regulate the access to the location information of a given vehicle and consists of two rules combined using permit-override combining algorithm. The first rule states that subjects working in a certain insurance company are allowed to perform a *GET* operation to retrieve location information, whereas the second rule is used to restrict the access to the traffic authority operating in the region in which the vehicle is passing through. Note that policies, being evaluated off-line, can only be used to verify constraints on static properties of the subjects and resources. To account for context-depended properties (e.g., location, current time), policies also include constraints that are returned along with the authorization token and verified at run-time time (see below).[1] For instance, in our example, the constraint of the first rule allows the insurance company to retrieve location information of the vehicle only when a given individual is driving the vehicle.

Off-line token generation: The authorization process starts with the off-line generation of an authorization token (top of Figure 3.4). The requesting party requests the authorization token from the authorization server to access a resource (1). The authorization server determines the permissions of the requesting party on the resource on the basis of the policies provided by the resource owner and generates an authorization token listing all permissions the requesting party has over the resource (2). The token is then sent to the requesting party (3). Figure 3.5 shows the authorization token generated by evaluating the access request against the policy. The token contains the *permissions* of the requesting party on the resource along with the validity period of the token. It is worth noting that the *constraint* specified in the policy is passed, together with the permissions, to the authorization token in order to prevent application overprivilege [32, 62]. This constraint is then verified at run-time to determine whether access should be granted.

Run-time token validation: When the requesting party wants to access a resource, she sends a request to the PEP located in the resource server along with the authorization token (4). The resource server verifies whether the token is valid (5). In addition, the resource server verifies the constraints provided in the token. If additional information is needed to verify the constraints on the context, the resource server retrieves it from the appropriate sources (PIPs), which can be located within the resource server or in a different component (6 and 7). For example, the resource server could be vehicle which may have to retrieve context information

[1] Constraints can be specified in XACML using element <Obligations>. In XACML, obligations are returned along with the access decision (either permit or deny) to enrich the decision.

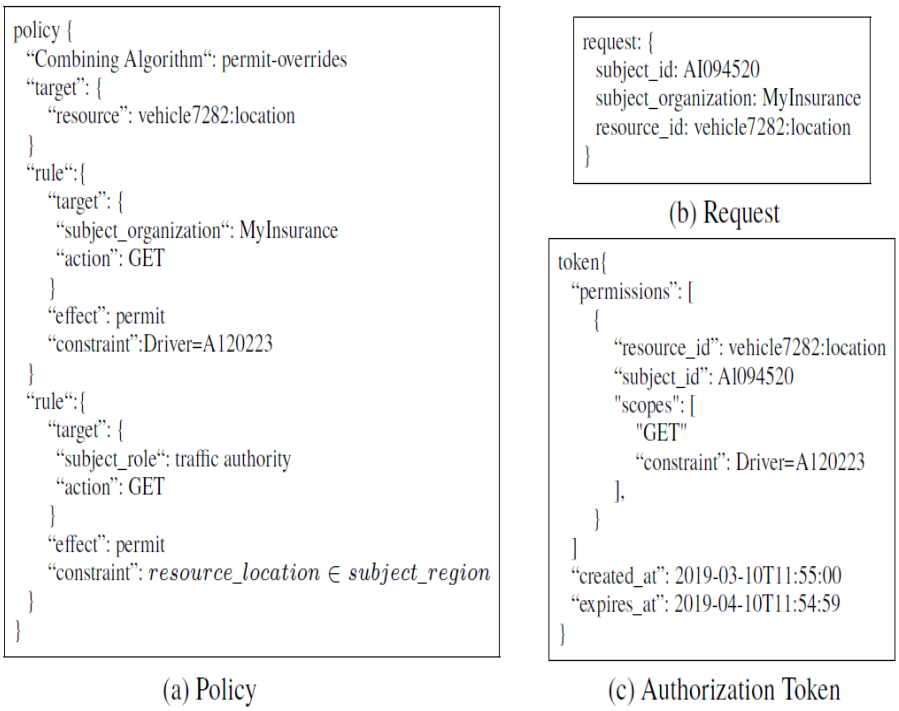


Figure 3.5: Example of policy, request and token

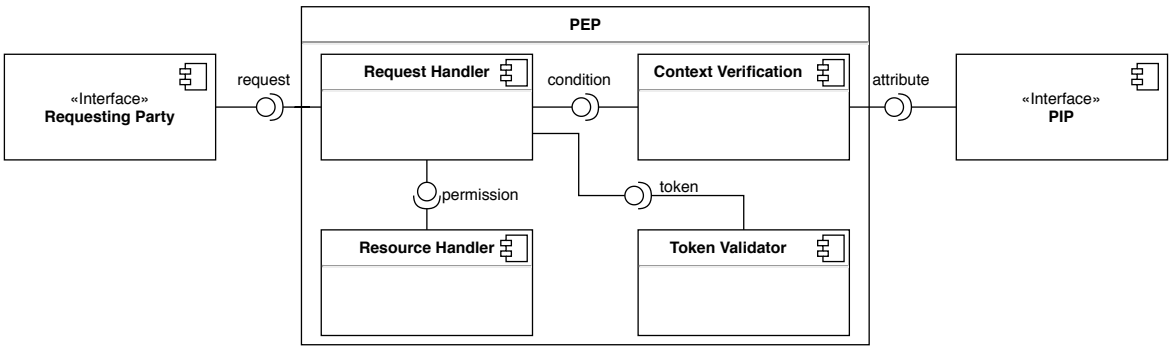


Figure 3.6: Component Diagram of PEP

from the nearby RSUs. If the verification of the constraint (8) is successful, the resource is disclose to the requesting party (9). Figure 3.6 provides a detailed view of the components and interfaces involved in the run-time token validation process.

Note that authorization tokens needs to be protected against tampering or relay attacks. How tokens can be protected against those attacks is out of the scope of this work and we refer to [21] for approaches commonly used to secure authorization tokens.

3.5 Application to Location Tracking Services

This section presents typical C-ITS use case scenarios and discusses how our authorization framework can be deployed to deal with such scenarios.

3.5.1 Location Tracking Services

Location information is an enabler for several services in the C-ITS domain [6]. For instance, location information can be used to increase vehicular safety (such as notification of nearby accident), tracking of stolen vehicle, pay-per-drive insurance, car sharing, toll payment, etc. To enable the retrieval of location information from a vehicle, the vehicle owner typically has to activate the forwarding of location information within the vehicle, including setting the time interval data are transmitted. Since this might generate a large amount of data, it is not ideal to forward the data to the requester directly. To this end, in our scenarios, we envision that data are transmitted to one of the C-ITS central systems (e.g., a Data Provider Back Office in the cloud), from which data can be retrieved when needed. Below we present typical use case scenarios relying on location tracking. These scenarios are an adaptation of the ones defined in the ETSI standard [6].

Scenario 1: Pay per drive insurance. Consider two sibyls, Alice and Bob, who co-own a car. They want to insure their car, but they would like different types of insurance. While Bob prefers a fixed premium, Alice wants a pay-per-drive insurance where the premium of the insurance policy is based on the kilometers traveled. In order to calculate the premium, the insurance company should be able to retrieve the location information from the vehicle when it is driven by Alice.

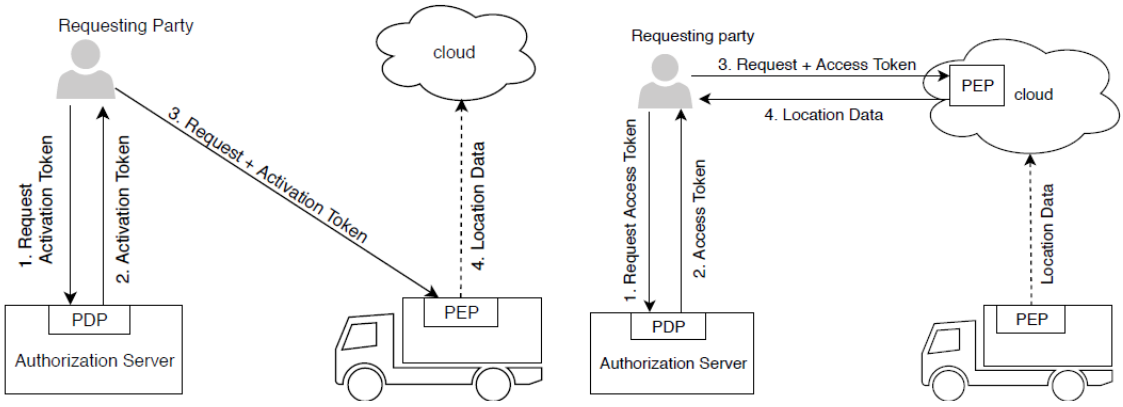
Scenario 2: Stolen Car. Alice and Bob's car was stolen and, thus, the two sibyls alert the police. Assuming that Bob has previously activated the forwarding of location information from the vehicle to the cloud, he can retrieve the exact location of his car in real time. Bob shares this information with the police to assist them in retrieving the car.

While enabling a variety of services, location information is sensitive and, thus, should be protected from unauthorized accesses. Next, we present how the authorization framework in Section 3.4 can be used to enable the selective sharing of location information.

3.5.2 Authorization Framework for Location Tracking Services

The first step for adapting our authorization framework to the scenarios above is to identify the C-ITS systems to which its components are deployed. In the scenarios, the owner of the vehicle represents the resource owner as he is the entity to whom information refers and, thus, he has the control on how the information is processed and to whom it can be disclosed [27]. The authorization server is handled by the Authorization Authority within the support system (cf. Fig. 3.2). The insurance company (scenario 1) and the police (scenario 2), which can be seen as two instances of the service provider back office (SP-BO) within the central system, are the requesting parties.

The scenarios involve two main phases: a first phase in which the forwarding of location information is activated and a second phase in which the information is retrieved from the Data Provider Back Office (DP-BO) that the vehicle owner used to store its data. Accordingly, the C-ITS system acting as the resource server varies in the two steps; in the first step the V-OBU acts as the resource server whereas in the second step the DP-BO acts as the resource server. We also distinguish two types of authorization tokens based on their purpose, namely *activation tokens* and *access tokens*. Activation tokens are used to enable the forwarding of location data from vehicle to the cloud. Access tokens are used to enable the retrieval of location information from the cloud.



(a) Activation of Location Data Forwarding (b) Retrieval of Location Data

Figure 3.7: Deployment of the authorization framework for location tracking

Forwarding of Location Information: Alice wants to activate the gathering of the location of her vehicle, e.g., to enable vehicle tracking as demanded by her insurance company. To this end, she enables the forwarding of location information from the vehicle to the cloud. Figure 3.7 depicts the forwarding activation process. The requesting party (acting on behalf of Alice) requests an activation token to the authorization server (1). The authorization server provides Alice with an activation token listing her permissions on the vehicle (2). These steps are performed during the off-line phase. At run-time, the requesting party provides the activation token to the V-OBU (3). The PEP in the vehicle validates the token as well as verifies the constraints on the context (if any). Upon successful validation, the vehicle starts forwarding location information to the cloud (4).

Retrieval of Location Information: Suppose that Alice has specified a policy that allow the insurance company to access location information of her car but only under the condition that she is driving (see Fig. 3.5). To comply with Alice's access requirements, the resource server (i.e., the cloud) has to verify this constraint at run-time before disclosing location data. This means that the resource server might have to retrieve additional information from the vehicle or road-side units in order to evaluate such constraints. Figure 3.7 depicts the information retrieval process. In the off-line phase, the requesting party (acting on behalf of the insurance company) requests an access token to the authorization server (1). The authorization server verifies the permissions of the insurance company and provides it with an access token (2). When requesting access to the location information of Alice's vehicle, the insurance company attaches the access token to the request (3). The resource server validates the token and verifies the constraints on the context conditions (i.e., whether Alice is driving). Upon successful validation, the location information is disclosed to the insurance company (4).

3.6 Discussion

This section discusses the feasibility of our framework and provides a qualitative analysis of the main design choices with respect to the challenges presented in Section 3.2. These choices

encompass the use of a hybrid authorization framework, the use of a centralized authorization server and the handling of contextual information.

We have adopted a hybrid authorization framework that combines principles of both policy-based and token-based frameworks. As discussed previously, policy-based frameworks perform policy evaluation at request time, introducing delay in service provisions. This, however, might be problematic in critical systems as C-ITS. In our design of the authorization framework, we leverage a token-based architecture where a token is generated off-line and then validated (along with the constraints on the context) at run-time, when access to a resource is requested. This allows performing policy evaluation off-line, thus reducing overhead and latency [43]. However, differently from existing token-based frameworks like OAuth [21], which require the resource owner to authorize an application the first time it requires access to a resource, we automate the generation of tokens by exploiting the use of policies. Although there have already been efforts to integrate the use of policies in the token-based architecture [4], existing framework usually do not support the verification of context conditions, making them unsuitable to deal with the dynamicity of C-ITS. It is worth noting that token validation along with verification of context conditions does not introduce a significant overhead as this operation is significantly less expensive than policy evaluation and token generation [30].

Our framework employs a centralized component for token generation (i.e., the authorization server). This provides resource owners with a single point for policy administration where they can efficiently manage their policies [8, 34]. The use of a centralized authorization server can also bring other advantages compared to deploying the policy decision point into (multiple) edge nodes (e.g. [58]) or within vehicles (e.g. [22]). For example, it allows exploiting the benefits of cloud computing in terms of scalability and reliability. It is worth noting that, in C-ITS, entities can rely on several resource servers to store and manage their data and resources. Therefore, an approach based on sticky policies [53], in which the resource server is required to attach policies to the data, is not particularly suitable as an entity would be required to configure their policies in each resource server in which her resources are stored.

In C-ITS, the information needed to verify context conditions may have to be retrieved from different sources, e.g. vehicles, road-side units or cloud. Thus, assuming that the resource server is the only source of context information as in [9, 28] restricts the types of context conditions that can be verified, thus limiting the level of granularity for access control. However, retrieving context information from different sources can have an impact on latency as it requires additional interactions between parties. Hence, one has to make a trade-off between the expressiveness of context conditions and the latency introduced by the retrieval of the information necessary for their verification.

Unlike other authorization frameworks, our framework has been designed to address the challenges characterizing the C-ITS domain. In this work, we have looked into these challenges from a design perspective. However, in practical deployments, other factors such as communication protocols (CoAP, MQTT) [37, 51], data format (JSON, XACML), handling of token refreshing and revocation, should be taken into account. Nevertheless, we believe that our hybrid authorization framework makes a step forward to the development of practical authorization mechanisms tailored to C-ITS. Moreover, the adoption of a C-ITS reference architecture as a baseline for our framework facilitates its integration and realization in existing C-ITS deployment sites.

4 Authentication

In 2013, Apple launched fingerprint recognition boosting security access to iPhones. The same technology becomes an enabler of several use cases in vehicles today such as driver authentication and identification, enabling new services beyond vehicle security. As a case in point, carsharing facilitates verification of different drivers sharing the same vehicles.

In this chapter, we introduce a use case where we deployed an authentication mechanism-based fingerprint and **OTP**(One Time Password).

4.1 Driver authentication

Driver authentication allows an access for owners or authorized drivers to their vehicles. This use case scenario enables other ITS services, such as car rental and driver profiling. It also requires a strong authentication mechanism. For this purpose we proposed an authentication system, taking into account several scenarios from connectivity perspective. In this document, we focus on the out-of-coverage driver authentication scenario. The work conducted during this period continues from the prior driver authentication use-case proposed and developed by KoçSistem. The latter proposed a driver authentication system relying on the driver fingerprint and **Kuksa** cloud process as shown in Fig.4.1, making the solution rely on continuous connectivity to authenticate drivers and authorize them to start the vehicle engine. It is worth noting that the proposed solution in this chapter ensures that authentication takes place in an out-of-coverage scenario.

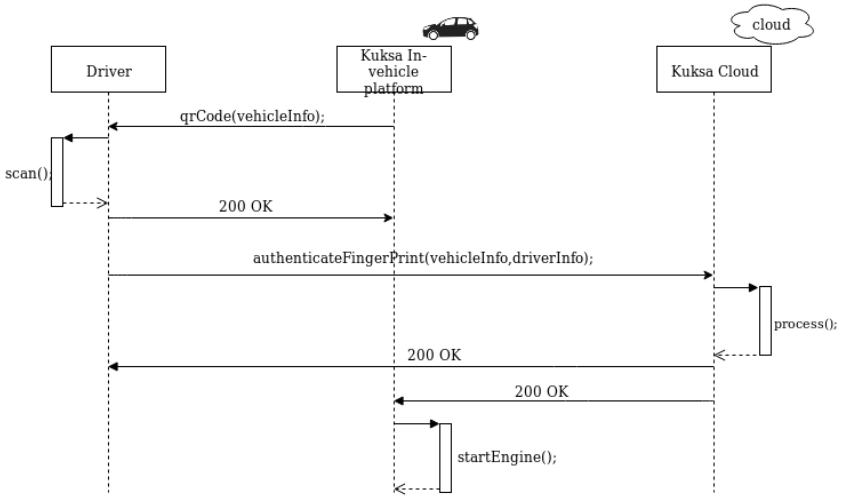


Figure 4.1: Authentication process in-coverage scenario.

4.1.1 Proposed solution

In this section, we describe the proposed solution and different technology enablers used to ensure the secure authentication process.

HMAC-based One-Time Password (HOTP) vs Time-based One-Time Password(TOTP)

- **HOTP**: Hash-based Message Authentication Code (HMAC) - based One-Time Password **OTP** is an initial One-Time Password algorithm which depends on two main types of information.

  The first information is a secret key, shared after terminating the driver registration as an owner of the vehicle, as shown in Fig.4.2. The secret is used to validate the submitted **OTP** for each authentication. It should be stored in a trusted zone of the vehicle hardware.

  The second information is a counter, which is the same value stored in the token and the server. The counter is incrementing in the token every time whenever a new authentication is performed or a token generation is necessary. For the server side, the counter is incremented only if the **OTP** is validated.

  In order to generate an **OTP**, the HMAC algorithm (SHA-1 hash) receives the counter via a token with the secret as the key.

- **TOTP**: Time-based **OTP** is HOTP-based, except that time is used instead of the counter. Each **OTP** stays valid for a given time, depending on the configuration (30 seconds or 60 seconds).

In the following sections, we explore and exploit the solution with TOTP as the algorithm of choice..

Discussion

In this section, we explore the proposed solution, as shown in Fig. 4.2. We identify two main processes instrumental to this solution - **Registration** as a black box and **Authentication** as a blue box:

- **Registration** takes place only during the initial phase. A driver starts a Two Factor Authentication (2FA), using a secure connection via the vehicle to the cloud. Once done, the driver should install an **OTP** application to their phone (Google authenticator or any other **OTP** app). The cloud sends an QR code thereafter, and is being displayed on the vehicle dashboard. The driver scans this QR code using the application they installed to receive a secret key immediately after successful vehicle registration. Our algorithm of choice here allows to generate an **OTP** token for every 30 seconds.

- **Authentication** is performed each time the driver wishes to start the vehicle engine. This process does not require connectivity to verify the driver identity. The driver sends an **OTP** token via a secure connection to the vehicle, abd if the token is valid, the engine will start.

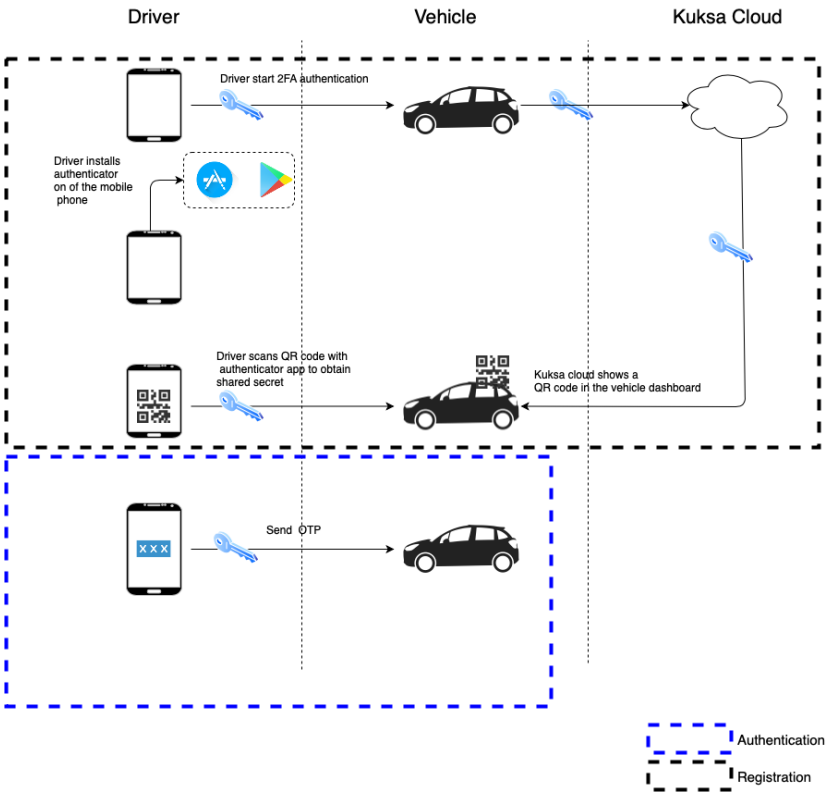


Figure 4.2: Flow authentication in out-of-coverage scenario.

As shown in Fig.4.3, the process starts by scanning a QR code with an **OTP** application installed on the phone. This step relates to the registration step, and takes places once for each vehicle when the driver sends an **OTP** token to start the vehicle engine. The received token is verified by the in-vehicle platform using the related algorithm thereafter.

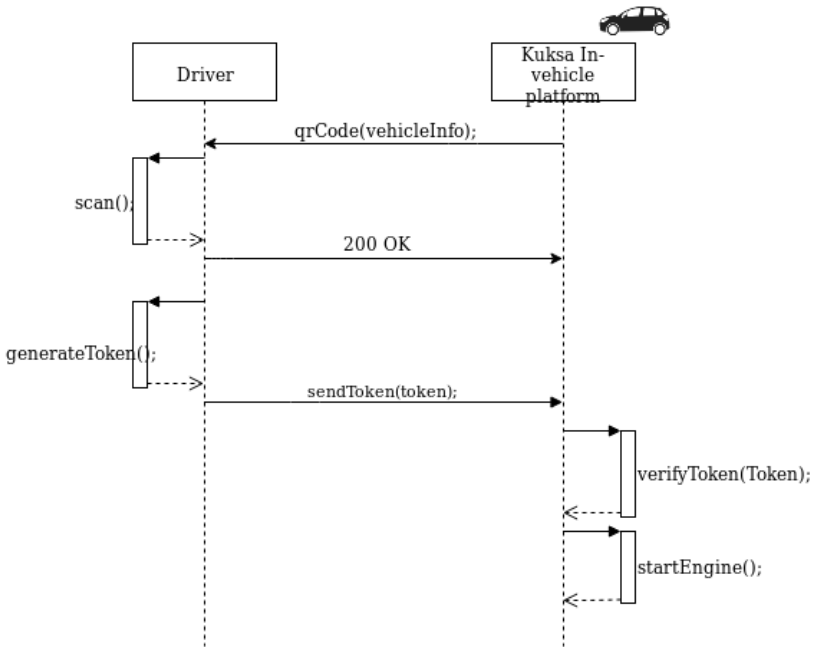


Figure 4.3: Authentication process in out-of-coverage scenario.

5 Conclusion

In this deliverable, we introduced different security implementations to secure connections between the vehicle and the cloud. We chose multiple use cases as proofs-of-concepts for these implementations. We proposed two IDSs to secure the in-vehicular networks, and introduced a deployed ITS-oriented authorization framework. Finally, we proposed an authentication mechanism for out-of-coverage scenario to make vehicle authentication efficient and adaptive.

Bibliography

[1] *CONVERGE*. https://converge-online.de. – Accessed: 2019-6-25

[2] *https://blog.paperspace.com/dimension-reduction-with-autoencoders/*. https://blog.paperspace.com/dimension-reduction-with-autoencoders/. – [Accessed 30-September-2019]

[3] *US-ITS*. https://local.iteris.com/arc-it. – Accessed: 2019-6-25

[4] *User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization*. https://kantarainitiative.org/file-downloads/rec-oauth-uma-grant-2-0-pdf/. – Accessed: 2019-6-25

[5] Directive 2010/40/EU of the European Parliament and of the Council. In: *Official Journal of the European Union* 50 (2010), P. 207

[6] Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management / ETSI. 2018 (102 940). – ETSI TS

[7] Abrougui, Kaouther ; Boukerche, Azzedine: Efficient group-based authentication protocol for location-based service discovery in intelligent transportation systems. In: *Sec Commun Netw* 6 (2013), Nb. 4, P. 473–484

[8] Ahmad, Tahir ; Morelli, Umberto ; Ranise, Silvio ; Zannone, Nicola: A Lazy Approach to Access Control as a Service (ACaaS) for IoT: An AWS Case Study. In: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2018, P. 235–246

[9] Albouq, Sami S. ; Fredericks, Erik M.: Securing communication between service providers and road side units in a connected vehicle infrastructure. In: *Proceedings of International Symposium on Network Computing and Applications*, IEEE, 2017, P. 1–5

[10] Amidi, Afshine ; Amidi, Shervine: A detailed example of how to use data generators with Keras. (2018). – [Accessed 08-March-2018]

[11] APPSTACLE: *D1.1: Specification of In-car Software Architecture for Car2X Applications*. January 2018

[12] APPSTACLE: *D1.3: Car2X Libraries, Interfaces and OTA Maintenance*. May 2018

[13] APPSTACLE: *D2.1: SotA Research with regard to Car2X Communication, Cloud and Network Middleware and corresponding Security Concepts*. January 2018

[14] APPSTACLE: *D2.2: Car2X Network and Connectivity Development*. February 2019

[15] Ashagoyaleduonix: Autoencoders in Keras. In: *Towards Data Science* (2019)

[16] Brownlee, Jason: How to Use ROC Curves and Precision-Recall Curves for Classification in Python. (2019)

[17] Chai, Tianfeng ; Draxler, Roland R.: Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. In: *Geoscientific model development* 7 (2014), Nb. 3, P. 1247–1250

[18] Chollet, François: *Keras Documentation*. `https://keras.io/`. – [Accessed 02-August-2018]

[19] Dajsuren, Y. ; Karkhanis, P. ; Kadiogullary, D. ; Fuenfrocken, M.: C-MobILE D3.1 Reference Architecture. URL `http://c-mobile-project.eu/library/`, 2017. – Research report

[20] Damen, Stan ; den Hartog, Jerry ; Zannone, Nicola: CollAC: Collaborative access control. In: *Proceedings of International Conference on Collaboration Technologies and Systems*, IEEE, 2014, P. 142–149

[21] Denniss, W. ; Bradley, J.: OAuth 2.0 for Native Apps / IETF. URL `https://tools.ietf.org/html/rfc6749`, 2017 (8252). – RFC

[22] Dorri, Ali ; Steger, Marco ; Kanhere, Salil S. ; Jurdak, Raja: Blockchain: A distributed solution to automotive security and privacy. In: *IEEE Communications Magazine* 55 (2017), Nb. 12, P. 119–125

[23] ETSI, TCITS: Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. In: *Draft ETSI TS* 20 (2011), P. 448–451

[24] ETSI, TCITS: Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service. In: *Draft ETSI TS* (2011)

[25] Evans, Lenny ; Ramasamy, Karthik: Exploring New Machine Learning Models for Account Security. In: *Medium* (2017). – [Accessed 08-October-2018]

[26] Festag, Andreas: Cooperative intelligent transport systems standards in Europe. In: *IEEE Communications Magazine* 52 (2014), Nb. 12, P. 166–172

[27] Guarda, Paolo ; Zannone, Nicola: Towards the development of privacy-aware systems. In: *Information & Software Technology* 51 (2009), Nb. 2, P. 337–350

[28] Gupta, Maanak ; Sandhu, Ravi: Authorization Framework for Secure Cloud Assisted Connected Cars and Vehicular Internet of Things. In: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2018, P. 193–204

[29] Han, Mee L. ; Kwak, Byung I. ; Kim, Huy K.: Anomaly intrusion detection method for vehicular networks based on survival analysis. In: *Vehicular communications* 14 (2018), P. 52–63

[30] Hernández-Ramos, José L ; Jara, Antonio J. ; Marin, Leandro ; Skarmeta, Antonio F.: Distributed Capability-based Access Control for the Internet of Things. In: *Journal of Internet Services and Information Security* 3 (2013), Nb. 3/4, P. 1–16

[31] Hobert, Laurens ; Festag, Andreas ; Llatser, Ignacio ; Altomare, Luciano ; Visintainer, Filippo ; Kovacs, Andras: Enhancements of V2X communication in support of cooperative autonomous driving. In: *IEEE communications magazine* 53 (2015), Nb. 12, P. 64–70

[32] Jia, Yunhan J. ; Chen, Qi A. ; Wang, Shiqi ; Rahmati, Amir ; Fernandes, Earlence ; Mao, Z. M. ; Prakash, Atul: ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms. In: *Proceedings of Network and Distributed System Security Symposium*, 2017

[33] Jordan, Jeremy: *Introduction to Autoencoders.* `https://www.jeremyjordan.me/autoencoders/`. 2018. – [Accessed 20-June-2018]

[34] Kaluvuri, Samuel P. ; Egner, Alexandru I. ; Hartog, Jerry den ; Zannone, Nicola: SAFAX - An Extensible Authorization Service for Cloud Environments. In: *Front. ICT* 2015 (2015)

[35] Karafili, Erisa ; Lupu, Emil C.: Enabling data sharing in contextual environments: Policy representation and analysis. In: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2017, P. 231–238

[36] Karkhanis, P. ; Brand, M. van den ; Rajkarnikar, S: Defining the C-ITS reference architecture. In: *Proceedings of International Conference on Software Architecture Companion*, IEEE, 2018, P. 148–151

[37] Laaroussi, Zakaria ; Morabito, Roberto ; Taleb, Tarik: Service Provisioning in Vehicular Networks through Edge and Cloud: an Empirical Analysis. In: *Proceedings of Conference on Standards for Communications and Networking*, IEEE, 2018

[38] Le, Van H. ; den Hartog, Jerry ; Zannone, Nicola: Security and privacy for innovative automotive applications: A survey. In: *Computer Communications* 132 (2018), P. 17–41

[39] Lee, Hyunsung ; Jeong, Seong H. ; Kim, Huy K.: Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)* IEEE (Organ.), 2017, P. 57–5709

[40] Liu, A. Y. ; Lam, D. N.: Using Consensus Clustering for Multi-view Anomaly Detection. In: *2012 IEEE Symposium on Security and Privacy Workshops*, 2012, P. 117–124

[41] Mahmudlu, Rauf ; den Hartog, Jerry ; Zannone, Nicola: Data Governance and Transparency for Collaborative Systems. In: *Data and Applications Security and Privacy XXX* Volume 9766, Springer, 2016, P. 199–216

[42] Markovitz, Moti ; Wool, Avishai: Field classification, modeling and anomaly detection in unknown CAN bus networks. In: *Vehicular Communications* 9 (2017), P. 43–52

[43] Martinez, Juan A. ; Ruiz, Pedro M. ; Marin, Rafael: Impact of the pre-authentication performance in vehicular networks. In: *Proceedings of Vehicular Technology Conference-Fall*, IEEE, 2010

[44] Matteucci, Ilaria ; Petrocchi, Marinella ; Sbodio, Marco L.: CNL4DSA: a controlled natural language for data sharing agreements. In: *Proceedings of Symposium on Applied Computing* ACM (Organ.), 2010, P. 616–620

[45] McKinney, Wes: pandas: a foundational Python library for data analysis and statistics. In: *Python for High Performance and Scientific Computing* 14 (2011)

[46] Miller, Charlie ; Valasek, Chris: A survey of remote automotive attack surfaces. In: *black hat USA* 2014 (2014), P. 94

[47] Mirsky, Yisroel ; Doitshman, Tomer ; Elovici, Yuval ; Shabtai, Asaf: Kitsune: an ensemble of autoencoders for online network intrusion detection. In: *arXiv preprint arXiv:1802.09089* (2018)

[48] Murugesan, Keerthiram ; Carbonell, Jaime ; Yang, Yiming: Co-clustering for multitask learning. In: *arXiv preprint arXiv:1703.00994* (2017)

[49] NXP: *Automotive Gateway: A Key Component to Securing the Connected Car.* – Available online: https://www.nxp.com/docs/en/white-paper/AUTOGWDEVWPUS.pdf

[50] OASIS ; Rissanen, Erik (Publisher): eXtensible Access Control Markup Language (XACML) v. 3.0. 2013. – OASIS Standard

[51] Ojanperä, Tiia ; Mäkelä, Jukka ; Mämmelä, Olli ; Majanen, Mikko ; Martikainen, Ossi: Use Cases and Communications Architecture for 5G-Enabled Road Safety Services. In: *Proceedings of European Conference on Networks and Communications*, IEEE, 2018, P. 335–340

[52] Paci, Federica ; Squicciarini, Anna C. ; Zannone, Nicola: Survey on Access Control for Community-Centered Collaborative Systems. In: *ACM Comput. Surv.* 51 (2018), Nb. 1, P. 6:1–6:38

[53] Pearson, Siani ; Casassa-Mont, Marco: Sticky policies: An approach for managing privacy across multiple parties. In: *Computer* 44 (2011), Nb. 9, P. 60–68

[54] Pedregosa, Fabian ; Varoquaux, Gaël ; Gramfort, Alexandre ; Michel, Vincent ; Thirion, Bertrand ; Grisel, Olivier ; Blondel, Mathieu ; Prettenhofer, Peter ; Weiss, Ron ; Dubourg, Vincent u. a.: Scikit-learn: Machine learning in Python. In: *Journal of machine learning research* 12 (2011), Nb. Oct, P. 2825–2830

[55] Ravi, Sujith: Graph-powered Machine Learning at Google. (2016). – [Accessed 10-July-2018]

[56] Ravidas, Sowmya ; Lekidis, Alexios ; Paci, Federica ; Zannone, Nicola: Access control in Internet-of-Things: A survey. In: *Journal of Network and Computer Applications* 144 (2019), P. 79–101

[57] Raya, Maxim ; Papadimitratos, Panos ; Hubaux, Jean-Pierre: Securing Vehicular Communications. In: *IEEE Wireless Communications* 13 (2006), Nb. 5, P. 8–15

[58] Riabi, Imen ; Saidane, Leila A. ; Ayed, Hella Kaffel-Ben: A proposal of a distributed access control over Fog computing: The ITS use case. In: *Proceedings of International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks*, IEEE, 2017

[59] Roermund, Fabrice P. Timo van: *Cybersecurity for ECUs: Attacks and countermeasures.* January 2017. – Available online: https://www.nxp.com/docs/en/white-paper/Cybersecurity-ECUs-WP.pdf

[60] Salonikias, Stavros ; Mavridis, Ioannis ; Gritzalis, Dimitris: Access control issues in utilizing fog computing for transport infrastructure. In: *Critical Information Infrastructures Security*, Springer, 2015 (LNCS 9578), P. 15–26

[61] Sambeek, M. van ; Ophelders, F. ; Bijlsma, T. ; Turetken, O. ; Eshuis, R. ; Traganos, K. ; Grefen, P.: Towards an architecture for cooperative-intelligent transport system (C-ITS) applications in the Netherlands / DITCM Innovations. 2015. – Research report

[62] Schuster, Roei ; Shmatikov, Vitaly ; Tromer, Eran: Situational Access Control in the Internet of Things. In: *Proceedings of Conference on Computer and Communications Security*, ACM, 2018, P. 1056–1073

[63] Sekar, Ramasubramanian ; Gupta, Ajay ; Frullo, James ; Shanbhag, Tushar ; Tiwari, Abhishek ; Yang, Henglin ; Zhou, Sheng: Specification-based anomaly detection: a new approach for detecting network intrusions. In: *Proceedings of the 9th ACM conference on Computer and communications security* ACM (Organ.), 2002, P. 265–274

[64] Seo, E. ; Song, H. M. ; Kim, H. K.: GIDS: GAN based Intrusion Detection System for In-Vehicle Network. In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug 2018, P. 1–6

[65] Sha, Kewei ; Xi, Yong ; Shi, Weisong ; Schwiebert, Loren ; Zhang, Tao: Adaptive privacy-preserving authentication in vehicular networks. In: *Proceedings of International Conference on Communications and Networking in China*, IEEE, 2006, P. 1–8

[66] Song, Hyun M. ; Kim, Ha R. ; Kim, Huy K.: Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In: *2016 international conference on information networking (ICOIN)* IEEE (Organ.), 2016, P. 63–68

[67] Sucasas, Victor ; Mantas, Georgios ; Saghezchi, Firooz B. ; Radwan, Ayman ; Rodriguez, Jonathan: An autonomous privacy-preserving authentication scheme for intelligent transportation systems. In: *Computers & Security* 60 (2016), P. 193–205

[68] Valkov, Venelin: Credit Card Fraud Detection using Autoencoders in Keras — TensorFlow for Hackers (Part VII). In: *Curiousily* (2017)

[69] Vandeput, Nicolas: Forecast KPI: RMSE, MAE, MAPE & Bias. In: *Analytics Vidhya* (2019)

[70] Willmott, Cort J. ; Matsuura, Kenji: Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. In: *Climate research* 30 (2005), Nb. 1, P. 79–82