TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technology
Department of Software Science

Tomoya Tanaka 201617IVCM

# Developmemt of IDS/IPS specifically designed for ETSI ITS-G5-based V2X communication

Master Thesis

**Technical Supervisor**
Olaf Manuel Maennel
Professor

Tallinn 2022

# Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

| | | |
|---|---|---|
| Author: | Tomoya Tanaka | Tomoya Tanaka |
| | | ...................................... |
| Date: | May 16th, 2022 | (signature) |

# Annotatsioon

Vehicle to Everything (V2X) side võimaldab sõidukitel saada teavet väliskeskkonnast ja saavutada turvalisema sõidu. Juhid saavad ümbritsevas keskkonnas sündmusi ja muid sõidukeid tajuda teiste sõidukitega sõnumeid vahetades. Lisaks aitab V2X-side isejuhtivatel sõidukitel oma liikumist turvalisemalt juhtida. Kui aga isejuhtival sõidukil saadud teade sisaldab valet teavet, võib käivitada sõiduki väärkäitumine. ETSI töötab standardite ITS-G5 dokumenteerimise kallal, kus on määratletud V2X side ja selle turbehaldussüsteem. Sellegipoolest ei arvesta ETSI ITS-G5 standard V2X-side kasutamise turvalisusega autonoomsete sõidukite juhtimiseks. See lõputöö analüüsib küberturvalisuse riske, mis tulenevad teiste sõidukite/taristute sõnumite kasutamisest ETSI ITS-G5-põhises V2X-sides sõidukite autonoomseks juhtimiseks. V2X-sidet toetavate sõidukivõrkude riskianalüüsi on läbi viidud paljudes dokumentides. Käesolevas lõputöös käsitletakse aga ETSI ITS-G5 protokollivirnu unikaalseid omadusi, näiteks geograafilist adresseerimist, mis võimaldab sõidukitel edastada sõnumeid kindlasse piirkonda. Lisaks pakutakse selles lõputöös välja IDS/IPS, mis on spetsiaalselt loodud ETSI ITS-G5 V2X side jaoks, et maandada tuvastatud küberturvalisuse riske. IDS/IPS, nimega Mitvane, on välja töötatud avatud lähtekoodiga tarkvarana. Hindamine kinnitab, et väljatöötatud IDS/IPS suudab vähendada küberturvalisuse riske, mida on ainulaadselt täheldatud ETSI ITS-G5-põhises V2X-suhtluses.

Lõputöö on inglisekeelne ja sisaldab 126 lehekülge teksti, 9 peatükki, 29 joonist, 37 tabelit.

# Abstract

Vehicle to Everything (V2X) communication enables vehicles to obtain information from external environments and achieve safer driving. Drivers can perceive events and other vehicles in the surrounding environment by exchanging messages with other vehicles. Besides, the V2X communication helps self-driving vehicles control their motion more safely. However, if a message received by a self-driving vehicle includes wrong information, misbehavior of the vehicle could be triggered. ETSI is working on documenting standards called ITS-G5, where V2X communication and its security management system are defined. Nonetheless, the ETSI ITS-G5 standard does not consider security for utilizing V2X communication for controlling autonomous vehicles. This thesis analyzes the cybersecurity risks of utilizing messages from other vehicles/infrastructures on the ETSI ITS-G5-based V2X communication to control vehicles autonomously. Risk analysis of vehicular networks that supports V2X communication has been conducted in many papers. However, this thesis considers unique features of the ETSI ITS-G5 protocol stack, such as geographical addressing, which enables vehicles to transmit messages to a specific area. Besides, this thesis proposes an IDS/IPS specifically designed for ETSI ITS-G5 V2X communication to mitigate the identified cybersecurity risks. The IDS/IPS, named Mitvane, is developed as open-source software. Evaluation verifies that the developed IDS/IPS can reduce cybersecurity risks uniquely observed in the ETSI ITS-G5-based V2X communication.

The thesis is in English and contains 126 pages of text, 9 chapters, 29 figures, 37 tables.

# List of abbreviations and terms

| | |
|---|---|
| ETSI | European Communication Standard Institute |
| ITS | Intelligent Transportation System |
| IEEE | Institute of Electrical and Electronics Engineers |
| V2X | Vehicle to Everything |
| PKI | Public Key Infrastructure |
| BTP | Basic Transport protocol |
| PASTA | Process for Attack Simulation and Threat Analysis |
| P2P | Peer To Peer |
| VANET | Vehicular Ad-hoc NETwork |
| OSS | Open Source Software |
| RSU | Road Side Unit |
| ITS-S | ITS Station |
| CA | Cooperative Awareness |
| DEN | Decentralized Environmental Notification |
| RLT | Road and Lane Topology |
| TLM | Traffic Light Maneuver |
| CAM | Cooperative Awareness Message |
| DEN | Decentralized Environmental Notification Message |
| SPATEM | Signal Phase And Timing Extended Message |
| MAPEM | MAP Extended Message |
| GBC | GeoBroadCast |
| TSB | Topologically-Scoped Broadcast |
| SHB | Single-Hop Broadcast |
| OBU | On-Board Unit |
| DFD | Data Flow Diagram |
| CAN | Controller Area Network |
| CRL | Certificate Revocation List |
| DPD | Duplicate Packet Detection |

# Table of Contents

# List of Figures

# List of Tables

# 1. Introduction

Vehicle to Everything (V2X) communication enables vehicles to obtain information from external environments and achieve safer and more efficient driving. Vehicles can perceive events and other vehicles by exchanging messages even when the events and the other vehicles are not visible from the vehicles. The application achieved by V2X communication includes cooperative awareness and decentralized notification. The cooperative awareness enables vehicles to receive information such as abnormal behavior of vehicles and the existence of an emergency vehicle. On the other hand, decentralized environment notification conveys information such as the existence of a vulnerable pedestrian and an accident that happened on the road. The information can be used to help drivers to comprehend driving conditions. Besides, self-driving vehicles can be controlled autonomously using the information.

Although V2X communication has the potential to improve road safety significantly, it has inherent risks that cannot be ignored. A vehicle that receives V2X messages could change its behavior based on the information included in the messages. If the message received by a vehicle is malformed, including wrong information, misbehavior of the vehicle could be triggered. When the vehicle is controlled autonomously, not relying on the control of a driver, the risk becomes much more critical. Motion adjustments made by an autonomous controller based on the information obtained by sensing or V2X communication become larger as the autonomous level of a vehicles increases. In that sense, it is imperative to consider the security of V2X communication to prevent the misbehavior of autonomous vehicles.

One of the key challenges in V2X communication security is in verifying the authenticity and integrity of exchanged messages. ETSI (European Telecommunication Standards Institute) in Europe is a primary actor working on the standardization of services supported by V2X communication as well as the architecture of the base system called C-ITS (Cooperative Intelligent Transport System). The standard, ETSI ITS-G5, enables vehicles to exchange messages on P2P (Peer To Peer) networks called VANETs (Vehicle Adhoc Networks), where vehicles can communicate with each other directly. The ETSI ITS-G5 addresses the security problems by introducing security services and a security management system on top of PKI (Public Key Infrastructure). Authorities on PKI take responsibility for authenticating, authorizing and giving authorization tickets to vehicles that are participating in V2X communication. By sending a V2X message along with an authorization ticket, receivers of the message can verify the authenticity and integrity of the message.

However, the ETSI ITS-G5 standard does not consider security to utilize the information in exchanged messages for controlling autonomous vehicles. Therefore, the ETSI ITS-G5 V2X communication system could have vulnerabilities, especially when utilizing messages coming from other vehicles/infrastructures to control vehicles autonomously. Besides, several features of the ETSI ITS-G5 protocol stack make it easy to exploit potential vulnerabilities underlying the ETSI ITS-G5 standard. The protocol used in ETSI ITS-G5 allows broadcasting messages to a specific destination area and multi-hop forwarding. These features enable a malicious vehicle/person to send malformed messages to vehicles in a specific area, even when the area is far from the malicious vehicle/person. Therefore, the malicious vehicle/person can conduct attacks against vehicles in any areas without changing its position.

To detect and drop malformed messages, IDS/IPS (Intrusion Detection System / Intrusion Prevention System) can be an effective solution [1]. However, IDS/IPS specifically designed for the ETSI ITS-G5 standard has not been developed and is not publicly available. The ETSI ITS-G5 standard uses a unique networking layer and transport layer protocol called GeoNetworking and BTP (Basic Transport Protocol), while other V2X communication standards such as IEEE1609 use IP protocol and UDP/TCP protocol for network and transport layer [2]. Generic IDS/IPS such as Suricata [3] does not support the protocols unique to ETSI ITS-G5. Besides, IDS/IPS specifically designed for ETSI ITS-G5 is not proposed or developed in existing research.

This thesis analyzes the cybersecurity risks of utilizing messages exchanged in the ETSI ITS-G5 V2X communication system for controlling autonomous vehicles. Based on the evaluation, the thesis focuses on IDS/IPS and proposes the solution to mitigate the cybersecurity risks. The features of IDS/IPS are determined by the risks to be mitigated identified in the risk analysis.

## 1.1 Problem statements

The Problem Statements (PSs) are summarized as follows:

- *PS-1* ETSI ITS-G5 could have inherent cybersecurity risks such as notification of wrong information to utilize messages coming from other vehicles/infrastructures for controlling autonomous vehicle.

- *PS-2* IDSs/IPSs as a solution to mitigate the inherent cybersecurity risks of utilizing messages coming from other vehicles/infrastructures in the ETSI ITS-G5 V2X communication system for controlling vehicles autonomously are not publicly available nor proposed in existing research.

## 1.2 Research questions

The Research Questions (RQs) this thesis answers are:

- *RQ-1* What are the cybersecurity risks of utilize messages coming from other vehicles/infrastructures on the ETSI ITS-G5 V2X communication system to control vehicles autonomously?
- *RQ-2* What countermeasures are effective to mitigate the identified cybersecurity risks given as the answer of *RQ-1*?
- *RQ-3* What functions are required for IDS/IPS to realize countermeasures given as the answer of *RQ-2*?
- *RQ-4* How the IDS/IPS should be implemented to satisfy the requirements given as the answer of *RQ-3*?

## 1.3 Contributions

The contributions of this thesis are summarized as follows.

- This is the first thesis/paper that analyzes the cybersecurity risks of utilizing messages coming from external vehicles/infrastructures in the ETSI ITS-G5 V2X communication system to control vehicles autonomously. The analysis is conducted following PASTA (Process for Attack Simulation and Threat Analysis). In the PASTA, enumeration of threats against autonomous vehicles, which uses messages coming from external entities, and analysis of the weakness/vulnerabilities of the ETSI ITS-G5 V2X communication system are firstly conducted. Next, attack vectors are identified after arranging attack scenarios into attack trees. Finally, the cybersecurity risks of utilizing messages from other vehicles/infrastructures on the ETSI ITS-G5 V2X communication system to autonomously control vehicles are identified. Besides, countermeasures to mitigate the identified risks are proposed.
- This is the first thesis/paper that proposes an IDS/IPS specifically designed for ETSI ITS-G5 V2X communication. The IDS/IPS, named Mitvane, is developed as OSS (Open Source Software). First, the system architecture and required features of the

IDS/IPS are determined based on the countermeasures proposed in the risk analysis. Next, the required features are implemented and made publicly available.

## 1.4 Thesis structure

The remainder of the thesis is organized as follows. Chapter 2 explains related work to clarify the position of our research compared to the existing reesearch. Chapter 3 explains technology stacks that work as the components to exchange V2X messages between vehicles on the ETSI ITS-G5 communication system and utilize the V2X messages for controlling autonomous vehicles. Threat modeling and risk analysis to utilize messages exchanged on the ETSI ITS-G5 V2X communication system for controlling autonomous vehicles are performed in Chapter 4. Chapter 5 gives a detailed explanation of packet structure exchanged in ETSI ITS-G5 based V2X communication system. Based on the observation of packet structure, system architecture and required features of the proposed IDS/IPS are described in Chapter 6. Chapter 7 shows implementation of the proposed IDS/IPS. Finally, a security test is conducted in Chapter 8 to verify that the prioritized risks are mitigated.

# 2. Related work

## 2.1 Applications achieved by ETSI ITS-G5 V2X communication

Road safety, traffic management and infotainment are primary applications achieved by V2X communications [4]. Road safety applications help drivers to notice various potential dangers and situations, including ones that are not visible to the drivers [5]. Traffic management applications enable vehicles to drive efficiently by sharing information among the vehicles. Speed management and cooperative navigation are two typical groups of this type of applications [6].

As a V2X communication platform, ETSI (European Telecommunication Standards Institute) proposes C-ITS [7], which encompasses a basic set of applications for active road safety and traffic efficiency [8]. The ETSI provides standardization of protocol stack for V2X communication. The basis of the protocol stack is the IEEE802.11p standard, which is an amendment of IEEE 802.11 to support WAVE (Wireless Access of Vehicle Environments). Along with the IEEE802.11p medium access control (MAC) and physical layer (PHY) standard, IEEE 1609 family standards are used to enable the safety, mobility and security required for V2X communication [9].

With the messages standardized by ETSI or SAE J2735, various information can be notified to vehicles. Varga et al. proposed an ETSI ITS-G5 V2X communication-centric traffic light controller system with an elaborated algorithm, where signal states of traffic lights are sent from RSU (Road-Side Unit) to vehicles periodically [10]. They also implemented a testbed using OBU (On-Board Unit), RSU and actual vehicle, where the current signal states of nearby traffic lights are displayed in the vehicle. However, in most research work, the messages received at OBU are only used for such visualization, separated from the core self-driving process to control the vehicle. A primary example of the application is HMI (Human Machine Interface), which only visualizes events and warnings inside vehicles based on the information included in received messages.

Nevertheless, a few researchers work on enabling self-driving vehicles to utilize the messages exchanged in ETSI ITS-G5 V2X communication for object recognition and path planning. Tsukada et al. developed cooperative perception for autonomous vehicles named

AutoC2X [11]. AutoC2X sends information about detected objects to other vehicles and RSUs while receiving objects detected by other vehicles and RSUs. The AutoC2X enables vehicles to detect objects in a broader area even if the objects are in blind spots. Hirata et al. developed corporative planning, where future paths of multiple autonomous vehicles are shared at RSUs, and the RSUs compute coordinated paths based on the shared future paths so that the vehicles can drive efficiently [12]. Shan et al. demonstrated cooperative perception by developing an Intelligent Roadside Unit (IRSU) equipped with a Lidar sensor and a camera. The IRSU collects data from the sensors and disseminates it to nearby vehicles [13]. However, security aspects of utilizing exchanged messages in the V2X communication are not considered so much in all of the research work.

## 2.2 VANETs security

In ETSI standard, IEEE 1609.2 based security facilities secure data exchange [14]. With an implementation of this standard, it is guaranteed that messages are sent from a legitimate source. However, it is known that networks composed of vehicles and RSUs, generally called vehicular ad hoc networks (VANETs), are vulnerable to several attacks. In this section, possible attacks in VANETs and solutions to prevent the attacks are explained.

First of all, disseminating false information could cause the misbehavior of vehicles. For example, an attacker can manipulate other vehicles to take alternative roads to cause serious accidents. Kim et al. propose a message filtering model to detect bogus information, where the confidence level of a received message is calculated by combining data from various sources such as RSUs, local sensors and other vehicles [15]. The research introduced a reputation mechanism, where a reputation is assigned to each source. Based on the reputation values of information sources, the certainty of information included in received messages is calculated. Raya et al. presented Misbehavior Detection System (MDS) to detect attackers that are disseminating false information on top of PKI [16]. In the MDS, information in received messages is clustered by $k$-means clustering and detects abnormal information. Then, messages coming from the node which sent the abnormal information are blocked. Cao et al. proposed a method to determine whether the events received are correct or not through voting from vehicles that witnessed the events [17]. Perit et al. proposed a spoofed data detection mechanism, where a vehicle collects reports about a notified event from neighboring vehicles until the number of reports surpasses a certain threshold [18].

Next, Denial-of-Service (DoS) attacks could make functionalities of a system unavailable. Typically, the attackers send far more requests than the system can handle. In VANETs, an attacker could try to shut down the network established by vehicles and stop communication between vehicles and RSUs [19]. Soryal et al. presented a solution to detect DoS attacks in IEEE 802.11. They introduced an adaptive threshold, the maximum rate of messages which any node can send over time to another node [20]. If it is observed on the link-layer of IEEE 802.11 that messages from a node exceed the maximum rate of messages, the node is tagged as an attacker. Verma et al. devised a system to prevent DoS attacks by monitoring TCP packets. This system monitors SYN and the corresponding SYN-ACK packets and maps them. Kerrache et al. developed a framework called TFDD, where trust is established between vehicles to detect DoS and DDoS attacks [21]. In this framework, each vehicle holds honesty and quality weights for each neighboring vehicle. The honesty weight and quality weight are determined by the number of packets and the quality of packets received from a neighboring vehicle, respectively. Based on the honesty weight and quality weight, whether the node is conducting a DoS attack and whether the node can be trusted are determined.

Authentication scheme using private-public key pair is also an effective solution to prevent DoS from untrusted node [22]. Chin et al. proposed a decentralized authentication scheme called the Trusted-Extended Authentication Mechanism (TEAM) to achieve short-time private-public key-based authentication [23]. TEAM only uses lightweight operations to reduce the computational cost of key agreement process and authentication process. Even if IEEE 1609.2 is implemented, it is possible for attackers to broadcast messages with invalid signatures, which results in unnecessary signature verifications. To prevent DoS attacks against signature-based authentication, He et al. proposed a pre-authentication process before signature-based authentication [24].

Sybil attack was firstly introduced in 2002, by which attackers can bypass security mechanisms using multiple identities [25]. Golle et al. proposed a heuristic mechanism to detect inconsistencies between received data and its identity by comparing received data to the knowledge about the VANET. The knowledge about VANET is obtained by sensors such as cameras [26]. Xiao et al. proposed a Sybil attack detection scheme based on the estimated position of a node by analyzing signal strength distribution, where surrounding vehicles claim their own positions and the positions are verified by the strength of signal emitted from the vehicles [27]. With the knowledge about surrounding vehicles, messages from Sybil nodes can be ignored.

More recent solutions to prevent Sybil attack utilizes certificate or signature-based authentication. Lee et al. presented a protocol called DTSA (Detection Technique against a Sybil Attack), where each vehicle registers its unique ID to authority and obtains a validated anonymous ID and certificate [28]. In the DTSA, vehicles send messages to other vehicles with the anonymous ID and the certificate. Then, vehicles that received the messages ask the authority whether the certificate is valid or not. This mechanism is similar to the security management system of ETSI ITS-G5-based V2X communication. The security management system is described in Section 3.1. Rahbari et al. proposed a PKI-based authentication and message validation system [29]. In the PKI, hierarchical authorities consisting of local CA and home CA are deployed. Feng et al. suggested a system called Event-Based Reputation System (EBRS), where each vehicle has a public key and pseudonyms, which are valid for a limited time and validated by the Trusted Authority (TA) over RSUs [30]. The generated certificate by the public key as well as the pseudonyms are stored at RSU and used to verify received messages by vehicles after accepting the validation request from the vehicles. Besides, the trusted authority takes responsibility for managing the pseudonyms, which are valid only for a limited time. ETSI ITS-G5 based V2X communication system also uses pseudonyms with a limited valid time, and the pseudonyms are managed by Enrollment Authority (EA) [31].

However, none of the solutions are specifically designed for the ETSI ITS-G5 V2X communication system. The solutions are designed for generic VANETs, where the unique features of the ETSI ITS-G5 V2X communication system, such as the broadcasting of packets to a specific destination area, are not considered. The broadcasting to a specific area enables a malicious vehicle/person to send malformed messages to vehicles in a specific area, even when the area is far from the malicious vehicle/person. New vulnerabilities, attack vectors, and solutions should be discovered when considering such unique features of the ETSI-G5 V2X communication system.

## 2.3 IDS/IPS for VANETs

IDS (Intrusion Detection System) and Intrusion Prevention System (IPS) can also be effective solutions to prevent a set of possible attacks in VANETs, dealing with malicious activities of nodes. Primary detection modes of IDS include signature-based detection mode, watchdog-based detection mode and anomaly-based detection mode [32]. In this section, intrusion detection systems for each mode proposed in previous research work are explained.

Signature-based Intrusion Detection System (IDS) detects malicious nodes and attacks generated by the malicious nodes by matching data included in the received packets against the predefined database of signatures. In case of a match, an alert is generated, or the packet is blocked by an intrusion prevention mechanism. Sedjelmaci et al. presented a signature-based IDS called ELIVE (Efficient and Lightweight IDS for VANET) [33]. This IDS addresses three types of attacks, namely the Sybil attack, black hole attack and false alert notification attack. The black hole attack is one of the variants of DoS attack, where attacker drops received packets instead of forwarding them [34]. By the existence of the black hole node, vehicles cannot receive messages which cannot be delivered via the black hole node. The ELIVE proposed in [33] uses a set of signatures to detect malicious messages. For example, every time when a message is received, ELIVE verifies that the behavior of the vehicle which sent the message is not abnormal to detect false alert notifications. Alerts such as EEBL (Emergency Electric Brake Lights) and Post crash Notification(PCN) have a specific expected behavior [35]. For example, vehicles who alert its own emergency break by sending EEBL must slow down. The criteria to determine whether the vehicle is abnormal or not is given by the rules (signatures). Tomandl et al. proposed an IDS called REST-Net, which also verifies notified alerts by comparing the information included in the alerts and the behavior of the vehicle which sent alerts [36]. If the information included in the alert is not compatible with the actual behavior of the vehicle which sent the alert, the alert is dropped. In [37], a signature-based intrusion detection method that verifies vehicle movement by applying the observed vehicle's movements to a plausibility model. The plausibility model is described by a set of signatures. With their IDS, a fake vehicle can be identified because the movement of the fake vehicle cannot be observed and follow the plausibility model.

Watchdog-based IDS assigns tasks of monitoring other nodes to several nodes and detecting abnormal behavior. Rulareliya et al. proposed an IDS based on a watchdog mechanism for the detection of nodes performing malicious activities, where every node is equipped with a watchdog component for the monitoring and verification of neighboring nodes to check whether they forward the packet to the next node or not [38]. The IDS proposed by Dias et al. determines a reputation value of a node by cooperative exchange of reputation values of other nodes. Then, packets from nodes with a reputation value lower than a threshold are dropped by the IDS. Khan et al. proposed an algorithm that improves the quality of malicious node detection by selecting optimal verifiers for the node which has been identified as malicious [39]. Ruj et al. used a fine-based strategy to prevent false location and false alert injection attacks, where nodes identified as malicious are imposed fine from CA and discourage the node from sending malformed messages.

Anomaly-based IDSs compare the activity of the system with the normal behavior model of the system, where any events deviating from normal behavior are considered malicious, and alerts are generated [1]. This approach is capable of dealing with newer and out of rule attacks. Wahab et al. proposed an anomaly based IDS solution named CEAP, where SVM (Support Vector Machine) is used for identification and categorization of vehicles on the road as benign or malicious [40]. Zaidi et al. proposed an IDS which uses statistical approach and detection model based approach for the detection of false information attacks, where detection model are trained at every node so that any deviations from that can make the IDS to generate alarms [41]. Kang et al. proposed Deep Neural Network (DNN) based IDS for the detection of malicious nodes [42]. This IDS uses DNN to improve detection accuracy compared to the accuracy obtained by traditional machine learning approach. Hybrid-based intrusion detection systems utilize the power of signature-based detection and anomaly-based detection.

However, none of the research work proposed IDS/IPS to mitigate risks caused by unique features of the ETSI ITS-G5 V2X communication system, such as geographical addressing, which enables vehicles to transmit messages to a specific area. Besides, when especially considering the protocol stack of ETSI ITS-G5, the protocol stack specification of ETSI ITS-G5 can be exploited to develop an IDS/IPS.

# 3. Preliminaries

## 3.1 ETSI ITS-G5 V2X communication system

The primary components of the ETSI ITS-G5 V2X communication system include wireless communication technology on IEEE 802.11p, services responsible for messages exchange, the ETSI ITS-G5 protocol stack and security services that support the exchange of messages. From here, brief descriptions of the system components and application components are provided in the following order.

1. IEEE802.11p
2. services responsible of message exchange
3. ETSI ITS-G5 protocol stack
4. security services that support message exchange
5. autonomous vehicles' core process

In this section, all nodes that can participate in V2X communication are called ITS-S (ITS Station). The term ITS-S is found in a lot of ETSI standards such as [43].

IEEE802.11p enables ITS-Ss to form an ad-hoc P2P network consisting of multiple vehicles and roadside devices such as RSU. ETSI ITS-G5 communication system has the option to use a cellular network in addition to the ad-hoc network. However, only the P2P ad-hoc P2P network is focused in this thesis.

The services responsible for messages exchange focused in this thesis are CA (Cooperative Awareness), DEN (Decentralized Environmental Notification), RLT (Road and Lane topology) and TLM (Traffic Light Maneuver). The description of the services is available in Table 1.

The V2X communication protocol stack is standardized as shown in Figure 1 (derived from [47]). The protocol stack follows the OSI model, which means that the received packets are parsed and controlled by each layer. The network protocol in the V2X communication protocol stack is Geonetworking, whereas the transport protocol is BTP. On top of that, V2X messages such as CAM, DENM, SPATEM and MAPEM are embedded.

| **Service** | **Description** |
|---|---|
| CA | CA service enables road users and roadside infrastructure to be informed about each other's position and dynamics, where road users are all kinds of road vehicles like cars and trucks or even pedestrian and roadside infrastructure like RSU [44]. The information to be exchanged for cooperative awareness is packed up in the periodically transmitted Cooperative Awareness Message (CAM). |
| DEN | DEN service constructs, manages and processes the Decentralized Environmental Notification Message (DENM) [45]. The DENM contains information on events that happened in traffic environments, such as abnormal behavior of a vehicle and road hazards. |
| RLT | RLT service is an infrastructure service that manages the exchange of a digital topological map as MAPEM (MAP Extended Message), which defines the topology of an infrastructure area [46]. The MAPEM provides vehicles with the lane topology and allowed maneuvers within an intersection area or a road segment. |
| TLM | TLM service is an infrastructure service that manages the exchange of SPATEM (Signal Phase and Timing Extended Message) [46]. The SPATEM includes safety-related information for supporting vehicles to execute safe maneuvers in an intersection. The TLM service informs in real-time about the current signal state, the residual time of the state before changing to the next state. |

Table 1. *Services responsible for messages exchanges.*

The exchange of these messages between vehicles/infrastructures realizes safety and traffic efficiency applications like the focused V2X application, which utilizes the V2X messages for autonomous vehicles' core process. Security is provided along with the layered stack.

Among the protocols shown in the protocol stack, GeoNetworking has significance when explaining how V2X communication works. Therefore, a brief description of GeoNetworking protocol is provided here. GeoNetworking is a network protocol based on the usage of geographical positions for addressing [48]. The two significant features of the GeoNetworking are geographical addressing and geographical forwarding. The geographical addressing can be achieved by geographical position included in GeoNetworking header. By virtue of the geographical position, senders of a message can specify a region where the message should be delivered while receivers of the message can determine where the message comes from. Here, the range where the message can be delivered directly is restricted due to physical layer, which means that ITS-Ss out of the range cannot receive the packet, even though the ITS-Ss are in the destination area. The geographical forwarding solves this problem by enabling ITS-Ss to forward packets. Note that infinite packet forwarding is disabled by specifying a maximum hop limit.

The forwarding schemes include GeoUnicast, GeoBroadcast (GBC) and Topologically-

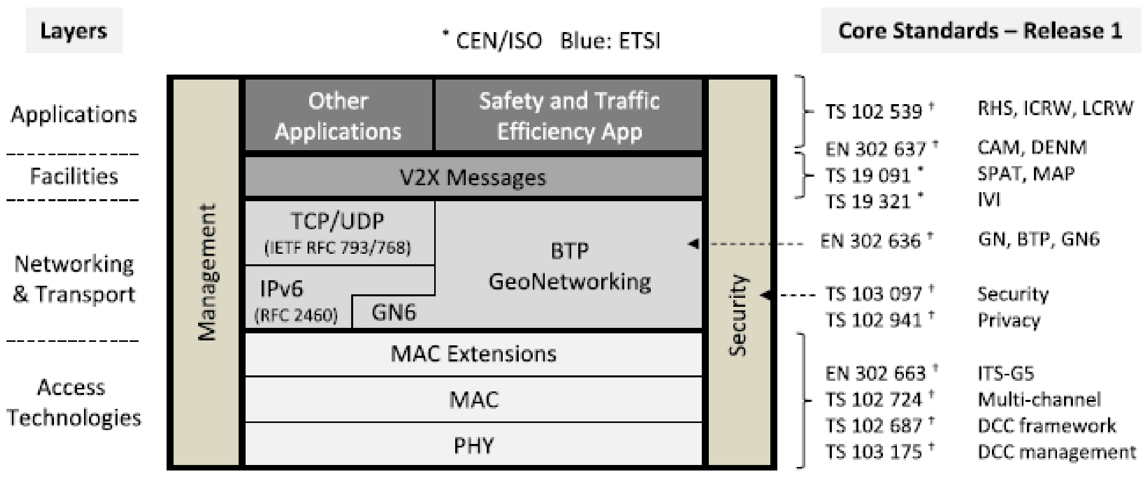


Figure 1. *ETSI ITS V2X communication protocol stack (derived from [47]).*

Scoped Broadcast (TSB) [48]. Figure 2, 3 and 4 shows a possible packet delivery in GeoUnicast, GBC and TSB, respectively. How the packet is delivered to other vehicles in each forwarding scheme is described below based on [48].

- **GeoUnicast:** When an ITS-S wishes to send a unicast packet, it first determines the destination's position. A receiver of the packet first sees the destination's position. If the position corresponds to the receiver's position, the receiver process the packet. Otherwise, the receiver forwards the data to an ITS-S towards the destination.

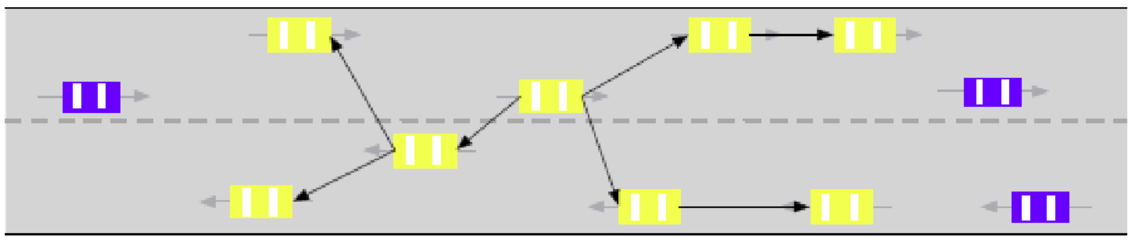


Figure 2. *GeoUnicast (derived from [48]).*

- **GeoBroadcast (GBC):** A sender of a packet has to determine an area where the message should be broadcasted. The shape of the area is a circle, rectangle or ellipse. Packets are delivered to the area in the same way as the GeoUnicast. The difference from GeoUnicast is that ITS-Ss in the destination area rebroadcast the packet. The number of permitted hops is restricted by the maximum hop limit configured to a packet. .

- **Topologically-scoped broadcast (TSB):** Packets also can be broadcasted topologically. The packets are delivered to all ITS-Ss which can be reached within a maximum hop limit. Single-hop broadcast (SHB) is a specific case of topologically-

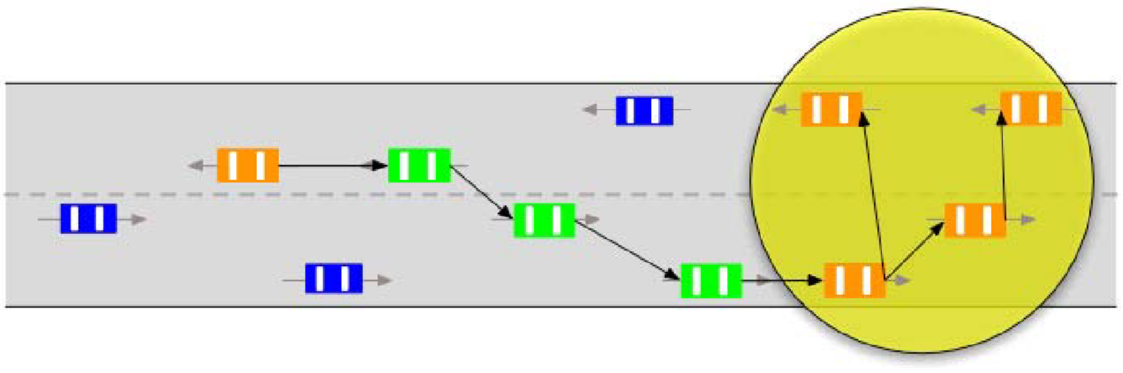


Figure 3. *GeoBroadcast (derived from [48]).*

scoped broadcast, which is used to send packets only to one-hop neighborhoods.

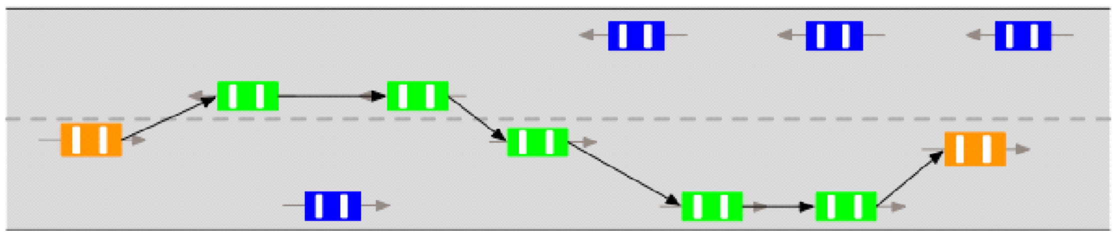


Figure 4. *Topologically-scoped broadcast (derived from [48]).*

In most use cases of the V2X communication system, GBC or TSB (SHB) is adopted. GBC is useful when a sender wants to transmit information to all ITS-Ss in a particular area. On the other hand, TSB is used especially for single-hop broadcast (SHB) by setting 0 to the maximum hop limit. By the SHB, the neighbors who can be reached by one-hop on IEEE802.11p wireless communication can receive the packet.

The security services in ITS-G5 are provided on a layer-by-layer basis, where the main layers which are supported by security services are the network layer and the facilities layer. In ETSI ITS-G5, the protocol for the network layer is GeoNetworking, while the facilities layer entities involve CA, DEN, RLT and TLM services.

All of the ITS services focused on this thesis, namely CA, DEN, RLT and TLM, must sign the communicated message by an authorization ticket. Note that encryption is not required in any cases. This means that attackers can see data included in V2X messages without any effort. Because V2X messages do not include confidential data, attackers cannot obtain personal information even if they sniff the messages. However, the messages shall have some identity to determine the origin of the message. To address this problem, ETSI defines a trust and security management system that provides pseudonymity and unlinkability [49]. Pseudonymity ensures that ITS-Ss can exchange V2X messages without disclosing their identity by which an ITS-S is uniquely determined. Unlinkability ensures that an ITS-S can transmit multiple messages while the multiple V2X messages are not correlated to the ITS-S.

Here is a brief description of the solution to provide pseudonymity and unlinkability in V2X communication. Each ITS-S shall have some identity to allow receivers of V2X messages to determine where the message comes from or who detects the event notified by the message. However, the identity should not be the one that can be attributed to a unique ITS-S. The identity uniquely assigned to each ITS-S is the canonical ID provided by a communication device manufacturer. However, the canonical ID cannot be included in the V2X messages because the canonical ID is attributed to a unique identity. Instead of the canonical ID, GeoNetworking address and station ID are used in V2X communications to identify the sender of a V2X message or the detector of an event. The GeoNetworking address and the station ID are temporal ones and managed by Enrollment Authority (EA). If EA receives a request from an ITS-S to change the temporal identity, a new identity is assigned to the ITS-S. This is the way to provide pseudonymity. Besides, because of the temporal features of IDs included in V2X messages, two different messages cannot be linked to one identity if the ITS-S properly changes its temporal in a fast cycle.

The security management system of ETSI ITS-G5 V2X communication highly relies on PKI (Public Key Infrastructure) [31]. Because of PKI based security management system, authentication, authorization, identity management, verification of messages can be achieved. The PKI architecture is shown in Figure 5.

First, an ITS-S obtains an enrollment credential from the Enrollment Authority (EA) after providing its canonical ID and public key. The canonical ID is an identity assigned by a manufacturer of the communication device that can be used to identify an ITS-S uniquely. The EA is responsible for authenticating ITS-Ss and managing the identities of ITS-Ss and their enrollment credentials. EA determines whether to permit an ITS-S to join a trusted network, where ITS-Ss can inform each other of the traffic environment by V2X communication. An ITS-S which could not obtain an enrollment credential cannot join the trusted network. Besides, the EA has the right to permit the use of specific services for ITS-Ss. For example, the EA can grant access for an ITS-S to only the DEN (Decentralized Environmental Notification) service and not the CA (Cooperative Awareness) service. Second, the ITS-S requests authorization certificates, also called authorization tickets, from Authorization Authority (AA). The AA provides the ITS-S with several specific permissions for each service allowed to use by EA, and the permissions are included in the

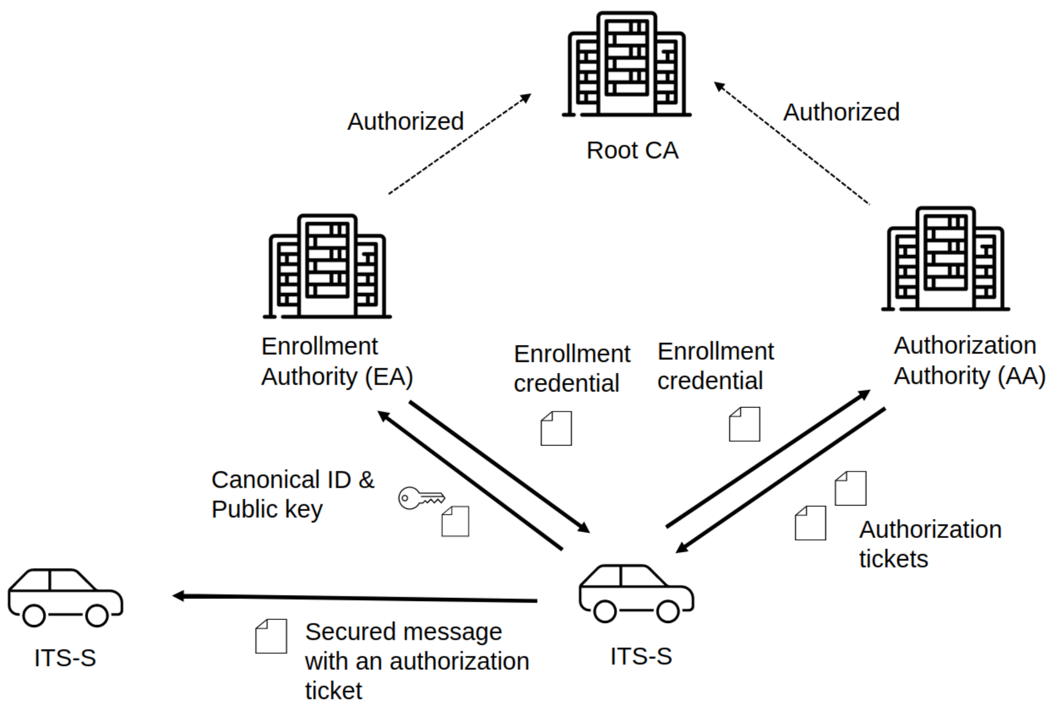

Authorized
Root CA
Authorized
Enrollment Authority (EA)
Enrollment credential
Enrollment credential
Authorization Authority (AA)
Canonical ID & Public key
Authorization tickets
ITS-S
Secured message with an authorization ticket
ITS-S

Figure 5. *PKI architecture.*

authorization tickets. For example, the permissions to be given by AA for the CA service include dissemination of public transport information and dissemination of emergency information. The EA and AA must also be authorized by a Root CA (Certificate Authority). Finally, the ITS-S signs payloads of V2X messages with the authorization ticket so that the other ITS-Ss can verify the validity of messages.

## 3.2 Core process of self-driving

Autonomous vehicles' core processes consist of several major processes, namely sensing, perception and decision [50]. Furthermore, each component can be divided into several tasks. The autonomous driving system technology stack is shown in Figure 6 (derived from [51]). The details of the autonomous vehicle technology stack are out of the scope of this paper. But, it is briefly described in order to understand how V2X messages can be utilized for the core process of autonomous vehicles based on [51].

In the sensing process, meaningful information is extracted from sensors. The sensors include GPS/IMU, LiDAR, and cameras. The GPS/IMU sensor is used to help autonomous vehicles localize themselves by reporting both inertial updates and a global position estimate at a high rate. LiDAR bounces a beam off surfaces and measures the reflection

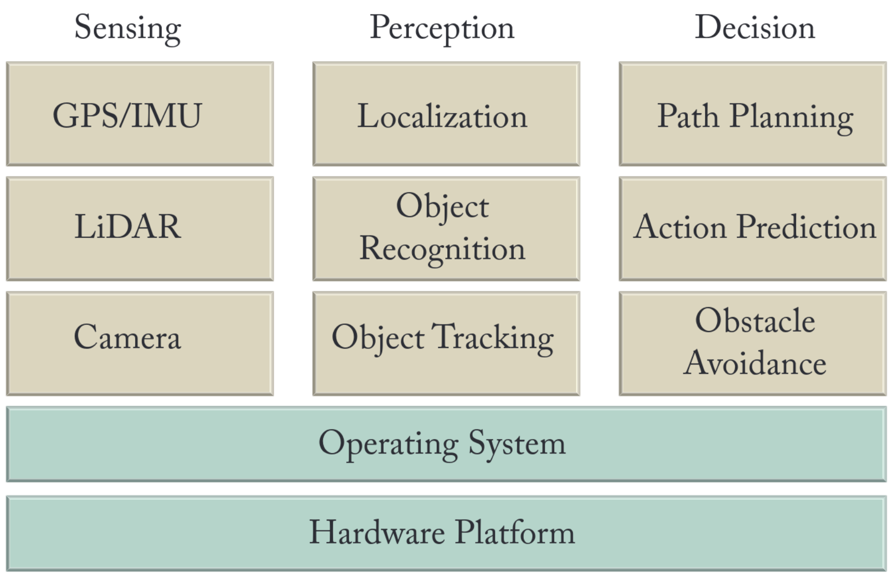


Figure 6. *Autonomous driving technology stack (derived from [51]).*

time to determine distance against objects in the surrounding environment. Cameras are mostly used for object recognition and object tracking.

Then, the perception process consumes incoming sensor data to understand the surrounding environment [50]. The main tasks include localization, object recognition and object tracking. In localization, the position of the ego-vehicle is determined using raw data from sensors. In object recognition, objects in the surrounding environment are detected, and sometimes labels are assigned to each object (*e.g.* this is a vehicle, this is a pedestrian). The data which can be used for object recognition come from LiDAR and cameras. Object tracking aims to estimate object states such as location, speed, and acceleration over time.

Based on the understanding of the vehicle's surrounding environment achieved by the perception stage, the decision stage can generate a safe and efficient action plan in real-time. The next path to be taken, for example, represented by a lane section, is determined in path planning. In action prediction, possible actions of other drivers which directly influence the ego vehicle's driving strategy are predicted. For the purpose of predicting the actions of other participants on the road, a stochastic model of the reachable position of the participants is generated, and probability distributions are associated with the reachable positions. Object avoidance prevents the ego-vehicle from colliding with vehicles, pedestrians and other obstacles. Typically, the object avoidance mechanism can be achieved by two approaches, namely a proactive approach and a reactive approach. In the proactive approach, measures like time to collision or predicted minimum distance are measured. After that, local path re-planning is performed based on the information. On the other hand, in the reactive approach, once sensors detect an obstacle ahead of the path, current

| **Name** | **Value** |
|---|---|
| Bandwidth | 10 MHz |
| Data Rates | 3 – 27 Mbps |
| Operating System | Linux 4.1.15 |
| Frequency Band | 5.9 GHz |
| GNSS | 2.5 m accuracy |

Table 2. *Cohda MK5 OBU Specs.*

control is overridden to avoid the obstacle.

## 3.3 Communication device

OBU (On-Board Unit) is a device used in the V2X communication system. OBU is installed on vehicles and sends/receives V2X messages on the IEEE802.11p channel. OBU for ITS-G5 is manufactured by several companies like Cohda Wireless and Huawei. The OBUs typically do not have high CPU specs, and the operating system is Linux. As an example, Table 2 shows several publicly available specs of Cohda MK5 OBU [52]. The CPU spec is not publicly available and is only stated in private documentation. Therefore, this cannot be described in this thesis, but a typical spec of OBU is a 32-bit single-core or dual-core ARM Processor with a 500 Mhz –1 Ghz clock.

## 3.4 Verification process of V2X messages

The verification process of V2X messages is involved in the security service of the network layer & transport layer. The router, the entity responsible for GeoNetworking functions, delegates the verification process to the security service for the networking & transport layer. After the verification process, the verification result is informed to the router. Figure 7 and Figure 8 shows the analysis of verification process for CAMs by flow chart diagram. Note that the verification process is described based on the specification in ETSI TS 103 097 V1.2.1 [53]. Several new versions are already available, and field names have been changed. However, basic functionality has not been changed from Version 1.2.1.

Once a V2X message is received and passed to the security service, the security service starts the verification process by finding possible certificates associated with the sender. If the V2X message includes a certificate or a certificate chain, which are indicated by the type of `SignerInfo` is `Certificate` and `Certificate_Chain`, respectively, there are no problems in finding possible certificates of the sender. Obviously, the possible certificate of the sender is the certificate included in the message. However, V2X messages sometimes do not include any certificates. Alternatively, the V2X messages include only the hash digest of the sender's certificate, which is indicated by the type of `SignerInfo` is `Certificate_Digest_With_SHA256`. In that case, the receiver of the message searches a certificate corresponding to the hash digest from a map. In the Figure 7, the map is represented by `cache_cert`. The `cache_cert` stores key-value sets, where a key is the hash digest of a certificate, and its value is the certificate object corresponding to the key. If a certificate chain is included in the V2X messages, the validity of AA's certificate is also verified regarding that the AA's certificate is consistent with the certificate of Root CA.

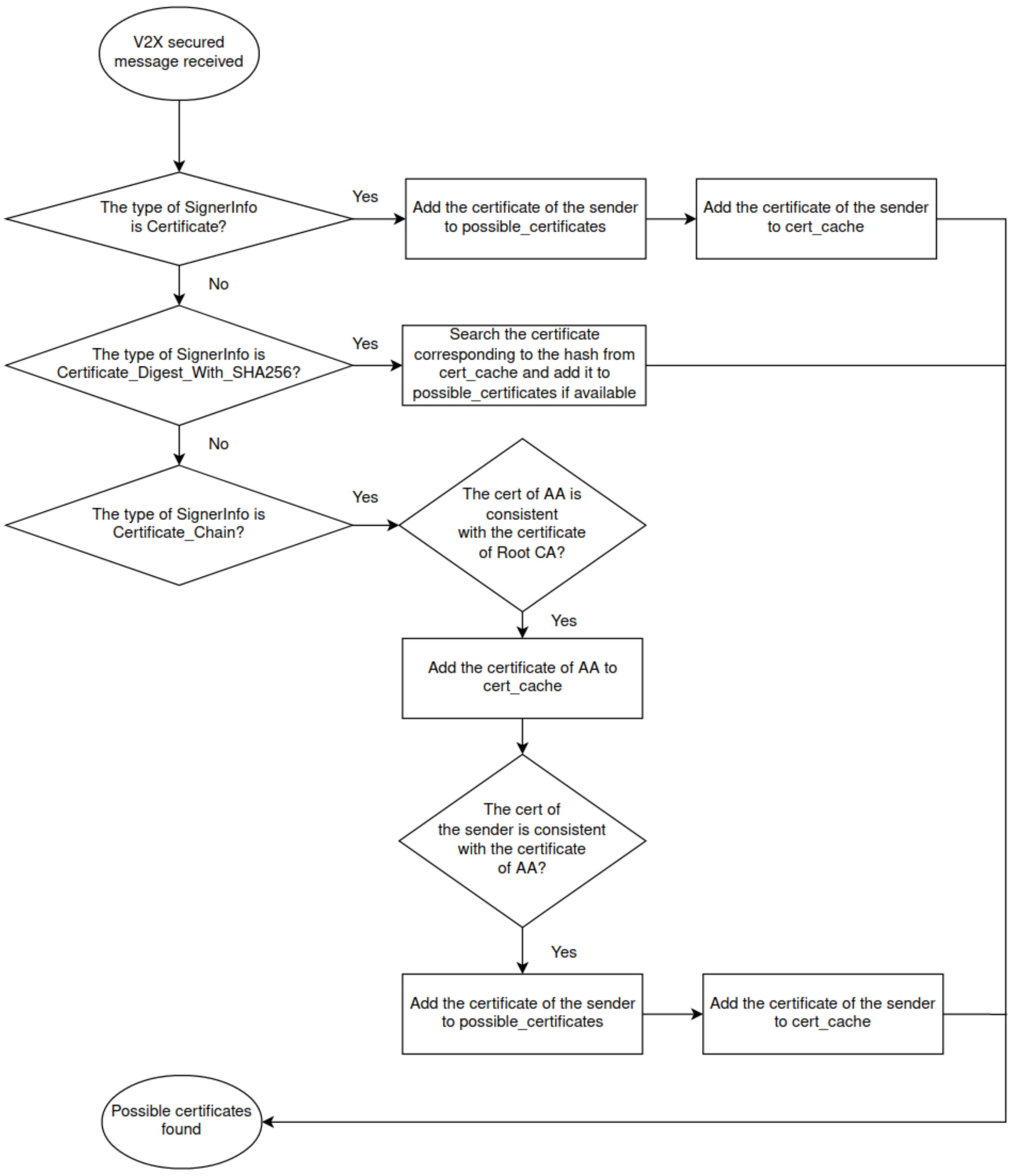

V2X secured message received
The type of SignerInfo is Certificate?
Yes
Add the certificate of the sender to possible_certificates
Add the certificate of the sender to cert_cache
No
The type of SignerInfo is Certificate_Digest_With_SHA256?
Yes
Search the certificate corresponding to the hash from cert_cache and add it to possible_certificates if available
No
The type of SignerInfo is Certificate_Chain?
Yes
The cert of AA is consistent with the certificate of Root CA?
Yes
Add the certificate of AA to cert_cache
The cert of the sender is consistent with the certificate of AA?
Yes
Add the certificate of the sender to possible_certificates
Add the certificate of the sender to cert_cache
Possible certificates found

Figure 7. *The process to find possible certificates.*

Next, validity of the found possible certificate is verified in the flow shown in Figure 8.

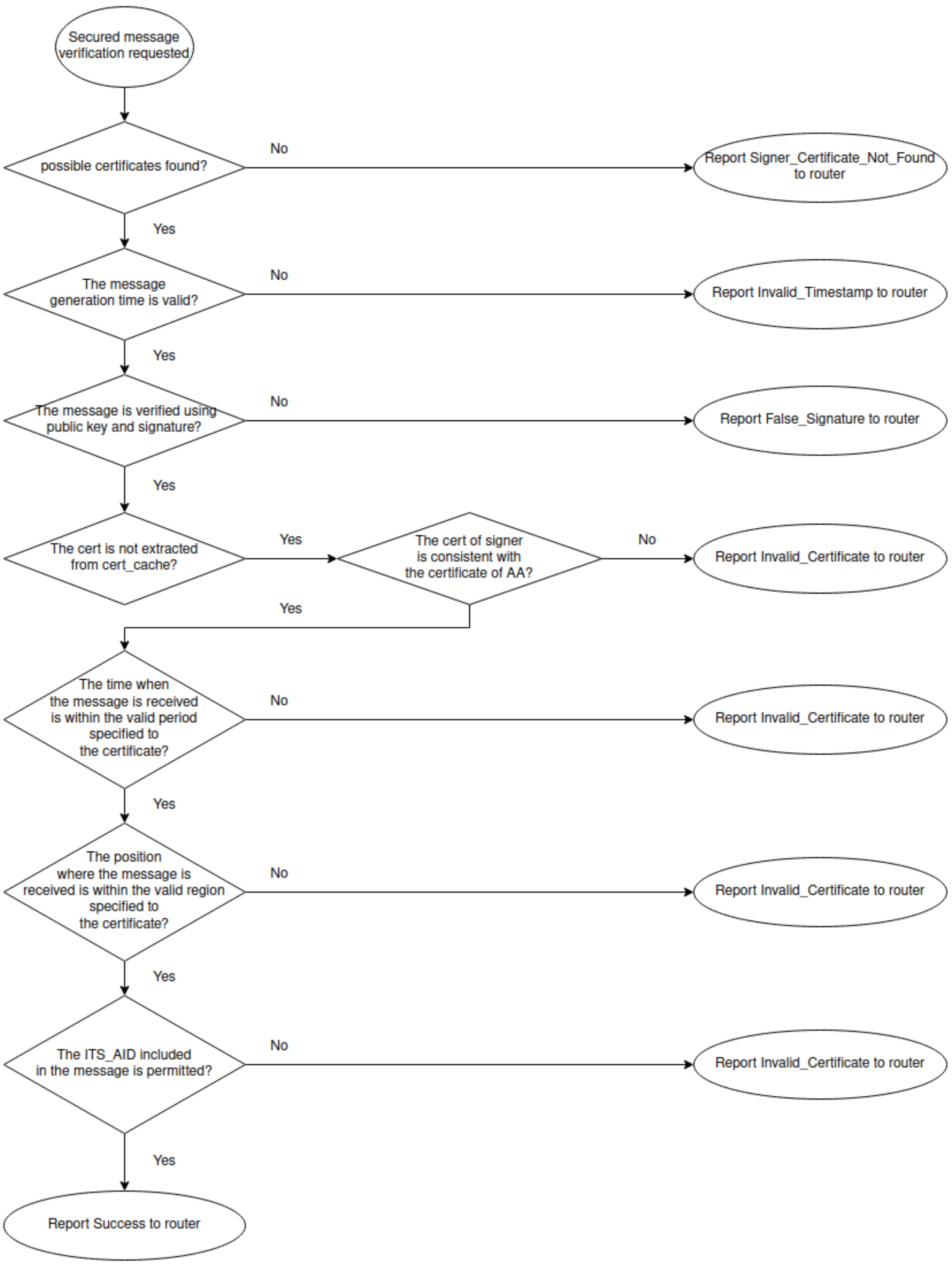
The verified attributes include

- Generation time
- Signature of the message
- Consistency with the signer of the certificate
- Consistency of the received time with the validity period
- Consistency of the received region with validity region
- Permission to access ITS Application

Generation time is the time when the secured message is created. How to validate the generation time is out of the scope of the ETSI standard, but according to C2C-CC Basic System Profile v1.1.0, the receiver should accept CAMs created in the last 2 seconds and other messages created in the last ten minutes [54]. Every secured message has a signature generated from the secure header in the GeoNetworking packet and message payload with the sender's private key. In the signature verification process, the signature is decrypted by the sender's public key included in the certificate and compared with the hash digest of the secure header and payload. In a similar way to signature verification, the consistency with the signer of the certificate is also verified using the signer's public key. The certificates always have a validity period. So, it shall be verified whether the time when an ITS-S receives a certificate is within the validity period. The validity region is optional, but it provides a region where the certificate is valid. The permission to use the ITS services is verified by checking `ITS_AID` field in the secure header. `ITS_AID` fields represent application IDs to which the sender of the message has access. For instance, if the received messages are DENM and the ID of DEN is not included in the `ITS_AID` field, the receiver should not consume the message.

The verification process of other V2X messages is not the same as the process for CAMs. One of the differences between CAMs and other V2X messages observed in the security header is that the type of `SignerInfo` is always `Certificate` in other V2X messages. In the technical specification ETSI TS 103 097 V1.2.1 [53], whether type of `SignerInfo` can be `Certificate_Chain` is not described. However, `Certificate_Chain` should be allowed because the signature of root CA or Authentication Authority (AA) is a significant factor that gives validity to the message. Therefore, it is assumed that the type of `SignerInfo` can be `Certificate` as well as `Certificate_Chain`. In Figure 7, the process of finding possible certificates for other V2X messages except for CAMs can be explained as follows:

- The type of `SignerInfo` is always `Certificate` or `Certificate_Chain`.
- The type of `SingerInfo` never becomes `Certificate_Digest_With_SHA256`.



Secured message verification requested
possible certificates found?
No
Report Signer_Certificate_Not_Found to router
Yes
The message generation time is valid?
No
Report Invalid_Timestamp to router
Yes
The message is verified using public key and signature?
No
Report False_Signature to router
Yes
The cert is not extracted from cert_cache?
Yes
The cert of signer is consistent with the certificate of AA?
No
Report Invalid_Certificate to router
Yes
The time when the message is received is within the valid period specified to the certificate?
No
Report Invalid_Certificate to router
Yes
The position where the message is received is within the valid region specified to the certificate?
No
Report Invalid_Certificate to router
Yes
The ITS_AID included in the message is permitted?
No
Report Invalid_Certificate to router
Yes
Report Success to router

Figure 8. *Secured message verification process.*

Besides, the validity region shall be included in other V2X messages while this field is not required for CAMs. This field contains the ITS-S's current location, where the contents of the security headers are generated. The certificate is valid when the position where the message is received is within the validity region.

## 3.5 Management of certificates

How to manage certificates in each ITS-S is not specified in ETSI standards. However, some management function is required to find the corresponding certificate to a hash digest if the type of `SignerInfo` is Certificate_Digest_With_SHA256. In this section, how certificates are managed in Vanetza [55], one of the ETSI ITS-G5 protocol stack implementations, is analyzed by a sequence diagram.

Figure 9 shows an initial certificate exchange process for CAMs when two ITS-Ss have not exchanged any messages yet. First, a sender sends a secured message to a receiver with a hash digest of the sender's certificate. The receiver tries to find a certificate corresponding to the sender's hash digest. However, the receiver cannot find the certificate because it is assumed that the two ITS-Ss have not communicated anything. Followed by the non-existence of the certificate in the receiver's `cert_cache`, the receiver sends back its certificate to the sender and requests the sender's certificate.

After the sender receives the certificate request, the sender sends back its certificate to the receiver. At this point, both the sender and the receiver obtain the other party's certificate, though AA's certificate has not been obtained. However, the AA's certificate can be obtained by the same process as the process to get the other party's certificate. The other party's certificate can be added to the `cert_cache` only after it is verified that the certificate is consistent with the AA's certificate.

After both the sender and the receiver have the other party's certificate and the corresponding AA's certificate in its `cert_cache`, the two ITS-Ss can exchange CAMs while verifying the messages. The sender sends a CAM with a hash digest of the sender's certificate, and the receiver gets the corresponding certificate from its `cert_cache`. The CAM has a signature that has been signed by the sender's private key. So, the message's validity can be verified by decrypting the signature with the sender's public key and comparing it to the hash digest of the CAM. Through this process, the integrity of the message is ensured.

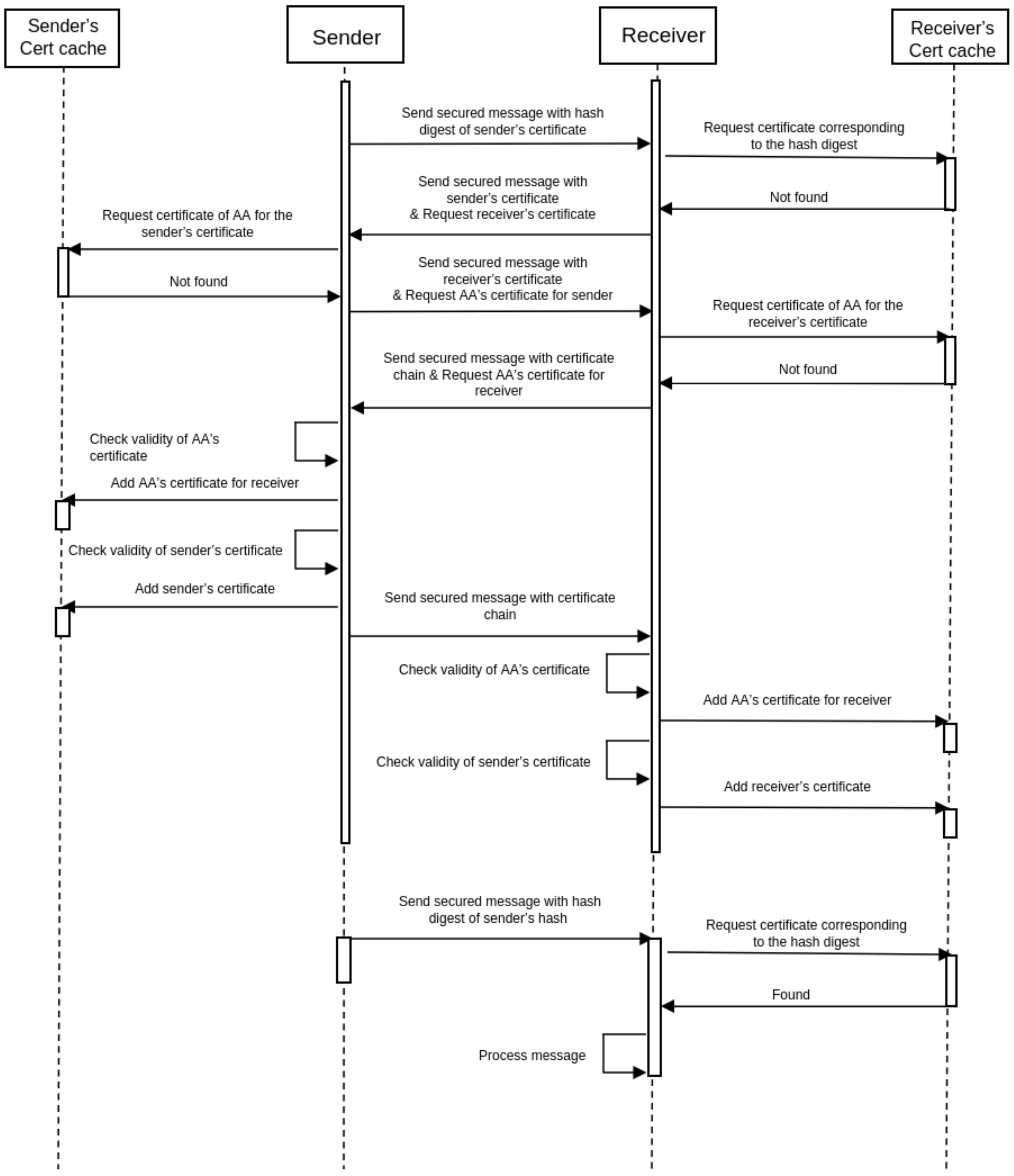

Sender's Cert cache
Sender
Receiver
Receiver's Cert cache
Send secured message with hash digest of sender's certificate
Request certificate corresponding to the hash digest
Not found
Send secured message with sender's certificate & Request receiver's certificate
Request certificate of AA for the sender's certificate
Not found
Send secured message with receiver's certificate & Request AA's certificate for sender
Request certificate of AA for the receiver's certificate
Not found
Send secured message with certificate chain & Request AA's certificate for receiver
Check validity of AA's certificate
Add AA's certificate for receiver
Check validity of sender's certificate
Add sender's certificate
Send secured message with certificate chain
Check validity of AA's certificate
Add AA's certificate for receiver
Check validity of sender's certificate
Add receiver's certificate
Send secured message with hash digest of sender's hash
Request certificate corresponding to the hash digest
Found
Process message

Figure 9. *Certificate management.*

# 4. Risk analysis

In this chapter, risk analysis is performed to identify weaknesses and possible attacks in the ETSI ITS-G5 V2X communication system when exchanging and utilizing V2X messages for core processes of autonomous vehicles such as object recognition and path planning.

The system this thesis focuses on is ETSI ITS-G5-based V2X communication system on VANETs. Hereinafter, the system is called "V2X communication system." The focused application which runs on the V2X communication system is the utilization of V2X messages for autonomous vehicles' core process. Hereinafter, the application is called "V2X application."

## 4.1 Methodology

The risk analysis or threat modeling methodologies include STRIDE, DREAD, Attack Tree, ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge), CAPEC (Common Attack Pattern Enumeration and Classification), and PASTA (Process for Attack Simulation and Threat Analysis) [56]. For the risk analysis of the V2X application running on the V2X communication system, the methodology has to satisfy the following requirements:

- Threats and risks can be analyzed beginning with the objective definition of the analyzed application.
- Technical components of the evaluated application can be analyzed in the risk analysis process.
- The weaknesses/vulnerabilities of the analyzed application should not be derived from well-known vulnerabilities, but should be derived from an analysis of technical components of the application and scope.
- The source of threat intelligence is not limited to a specific database.

The application addressed in this thesis has a particular high-level objective to be achieved: the utilization of V2X messages for autonomous vehicles' core process. The threat analysis should be rooted in the objective of the application. Vulnerabilities underlying the V2X application and V2X communication system are not organized as database such as CVE (Common Weakness Enumeration). Therefore, the vulnerabilities should be identified with detailed observations of the V2X communication system and the V2X application.

Among the mentioned methodologies, PASTA is chosen for the risk analysis methodology in this thesis. The PASTA is divided into the following seven stages [57, 58].

1. Definition of the application objectives
2. Definition of the technical scope
3. Application decomposition and analysis
4. Threat analysis
5. Weakness and vulnerability analysis
6. Attack modelling and simulation
7. Risk analysis and management

As the list of stages shows, the PASTA satisfies the requirements for the risk analysis of the V2X application running on the V2X communication system. In the application decomposition and analysis (stage 3), the technical components are analyzed in detail. In the threat analysis (stage 4), the threats are collected from various internal sources as well as external sources [59]. The weaknesses and vulnerabilities are analyzed based on the technical details of the application analyzed in stage 3. Nonetheless, STRIDE, DREAD and Attack Tree can be used for threat analysis (stage 4) and attack modeling and simulation (stage 6). As [59] adopted, the attack tree is used for attack modeling also in this thesis.

Note that the objectives to be defined in stage 1 is different from the high-level objective. The objectives to be defined in stage 1 are functional level objectives to realize the V2X application. In contrast, the high-level objective is a final goal that can be achieved after realizing all the functional level objectives defined in the stage 1. The high-level objective of the V2X application is the utilization of V2X communication for autonomous vehicles' core process. The functional objectives of the V2X application are defined in stage 1, considering the high-level objective. The following sections corresponds to the involved stages in PASTA. The risk analysis process is based on [59] and [58].

## 4.2 Definition of the objectives

The goal of this section is the definition of the functional level objectives, which is starting point to analyze threats and risks affecting the system later. The first step of stage 1 is the definition of the evaluated V2X application. The risk is analyzed for the defined application. The application can be defined by enumerating functions of the applications. In other words, the list of functions determines the behavior of the evaluated application.

The following is a list of Application Functions (AFs) for the V2X application that works along with the ETSI ITS-G5-based V2X communication system.

- *AF-1* A vehicle can send V2X messages to other vehicles using a dedicated device for ETSI ITS-G5 based V2X communication.
- *AF-2* A vehicle can receive V2X messages from other vehicles and RSUs using a dedicated device for ETSI ITS-G5 based V2X communication.
- *AF-3* A V2X message can be signed by a private key of the vehicle which generated the message.
- *AF-4* A vehicle can verify integrity and authenticity of received messages.
- *AF-5* A vehicle can utilize the received V2X messages for core processes of self-driving such as object recognition and path planning.
- *AF-6* A vehicle can apply for change of pseudonym which is used to identify itself in all communication with other vehicles.

The exchanged messages themselves should not be encrypted because the messages do not include PII (Personal Identifiable Information). Besides, the ETSI standard states that V2X messages shall be signed data type [44, 45, 46]. This implies that the messages do not need to be encrypted because the signed data type is chosen from the following 4 data types: raw data type, signed data type, encrypted data type and signed/encrypted data type. Therefore, encryption of data is not included in the AFs.

The next step is the definition of impacts to the system in case system assets are compromised. The system assets for the V2X application are vehicles and exchanged V2X messages. This activity consists of defining what impact will be made in the case of the system assets are compromised. The impacts are summarized as the followings:

- Loss of safety of vehicles
- Loss of availability of exchanged messages
- Loss of integrity of exchanged messages

The loss of availability of exchanged messages and loss of integrity of exchanged messages also results in loss of safety of vehicles. The listed losses are simple but sufficient in this stage. What threats causes the impacts are discussed in Section 4.8.

Finally, a risk profile of the V2X application is defined. The goal of this step is to provide a snapshot of the application focusing on inherent cybersecurity risks. Based on the risk profile, the scope of risk analysis is considered.

| Profile of the V2X application – utilization of V2X messages for core processes of autonomous vehicles | |
|---|---|
| General Description of the application functionality | The application allows autonomous vehicles to utilize received V2X messages for their core process such as object recognition and path planning. The V2X messages convey information about vehicles' state, events that happened in the surrounding environment, digital topological maps and the status of traffic signals. New vehicles can participate in the V2X communication after obtaining authorization tickets from an authority. V2X messages are signed by the authorization tickets and verified by the receivers. |
| Application type | Running on ETSI-G5 ITS communication system and facing a P2P network. |
| Data classification | Public, Non confidential, Non PII, Critical (Wrong data could cause other vehicle's misbehavior) |
| Information security risks | HIGH (High risk for potential loss of safety of vehicles, data integrity and data availability) |
| High Risk Transactions | Exchange of V2X messages. |
| User roles | Sender/receiver of V2X messages. |
| Number of users | Corresponds to the number of vehicles that support ETSI ITS-G5 communication. |

Table 3. *Risk Profile.*

## 4.3 Definition of technical scope

The goal of this stage is to enumerate the details of technical components/stacks which comprise the V2X application.

First of all, the participants involved in the V2X communication system for the V2X application are vehicles, pedestrians and infrastructures such as RSU (Road-Side Unit). RSU is a small server that stores static information such as lane topology and allowed maneuvers in a road section as well as collects information in a road section. Hereinafter, all the participants are lumped together and called ITS-S.

Next, application components for the V2X application are enumerated. The components in the V2X communication system are described in Section 3.1. In addition, the autonomous vehicle's core process that would utilize received V2X messages is also included in an application component. The autonomous vehicle's core process is described in Section 3.2.

The next step in Stage 2 is an enumeration of low-level components. This step aims to discover what types of devices and operating systems are being used. The device to be used in the V2X communication system for the V2X application is only OBU (On-Board Unit). The details of OBU are described in Section 3.3.

With the participants, application components and devices enumerated in this section, the architecture scope of the V2X application are depicted as shown in Figure 10 and Figure 11. Figure 10 shows scope in communication architecture among ITS-Ss. The ITS-Ss that can send messages to a vehicle are RSUs and other vehicles with OBU. V2X messages sent from other vehicles are DENM and CAM, while messages sent from RSUs are SPATEM and MAPEM as well as CAM and DENM forwarded by RSUs. A V2X message are packed into a GeoNetworking packet and transmitted into the IEEE802.11p channel. The forwarding schemes of GeoNetworking protocol are SHB, TSB or GBC shown in Figure 4 and Figure 3. Figure 11 shows internal architecture scope in an autonomous vehicle. First, received data is decapsulated, and the MAC header, GeoNetworking header, BTP header, and V2X message payload are obtained. The received V2X messages are temporally stored to LDM (Local Dynamic MAP) while sent to the computing unit connected with Ethernet. The computing unit performs perception and decisions utilizing V2X messages as well as sensor data. Finally, as the result of the decision, signals are sent over CAN (Controller Area Network), and the vehicle is actuated.

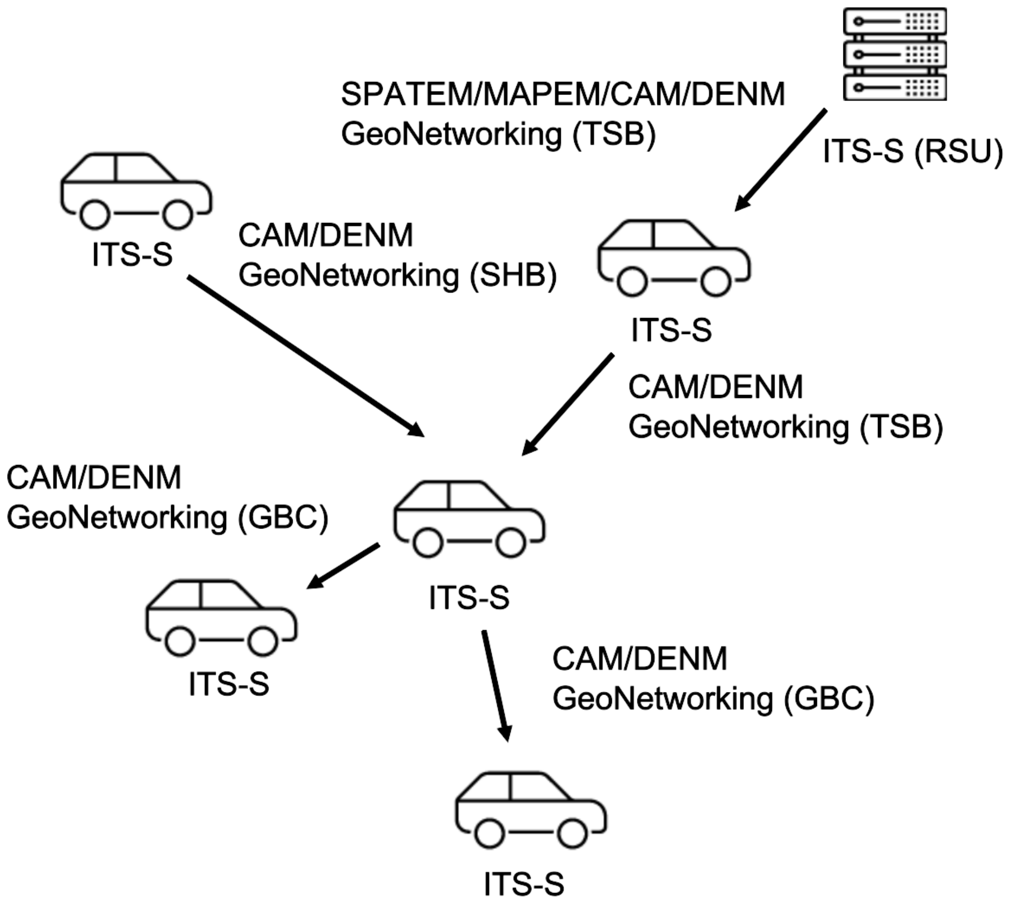


Figure 10. *Application Architecture Scope (Network).*

The dotted components are not the scope of the risk analysis in this thesis but are necessary

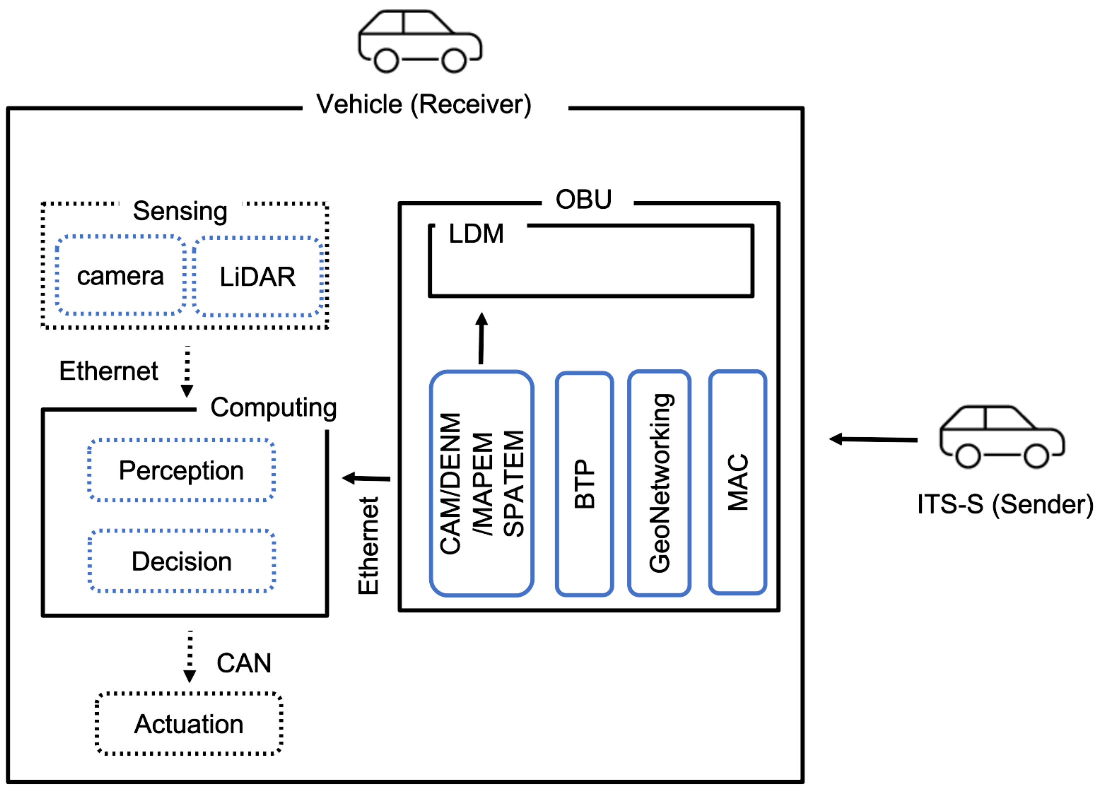


Figure 11. *Application Architecture Scope (Internal Processing).*

components for the V2X application. The dotted components are out of the scope because the objective of this risk analysis is to identify weaknesses and vulnerabilities in the ETSI ITS-G5 V2X communication system, as described at the beginning of this chapter. Sensing, computing, and actuation are not parts of communication. Therefore, they are not included in the scope.

## 4.4 Application decomposition and analysis

In this stage, the V2X application is decomposed into basic components, and security controls and interactions between these components are analyzed. The purpose of this stage is application decomposition to make it possible to analyze specific design flaws and vulnerabilities which the threat actors might seek to exploit.

The first step in this stage is the enumeration of use cases of the V2X communication system. It is assumed that the focused V2X application, utilization of V2X messages for the core process of autonomous vehicles, uses V2X messages generated and exchanged by the services, namely CA, DEN, RLT, and TLM. The enumerated use cases of the services are shown in Table 4 based on [14]. Then, autonomous vehicles use the information described in the use cases for the core process, such as perception and decision.

| Service | Use case |
|---|---|
| CA (Cooperative Awareness) | Generate and exchange emergency vehicle warning<br>Generate and exchange merging traffic turn collision risk warning<br>Generate and exchange intersection collision risk warning<br>Generate and exchange notification of Lane change manoeuver |
| DEN (Decentralized Environmental Notification) | Generate and exchange wrong way driving warning<br>Generate and exchange notification of stationary vehicle - accident<br>Generate and exchange notification of stationary vehicle - vehicle problem<br>Generate and exchange signal violation warning<br>Generate and exchange roadwork warning<br>Generate and exchange vulnerable road user warning |
| RLT | Generate and exchange notification of static geographic road information |
| TLM (Traffic Light Maneuver) | Generate and exchange current signal phase notification<br>Generate and exchange notification of the remaining time until the next phase change |

Table 4. *Use case enumeration.*

As the next step, Data Flow Diagram (DFD) in the application architecture scope defined in the Section 4.3 is created. The DFD makes it possible to visualize the flow of the information over trust boundaries in the V2X application. DFD for the V2X application is shown in Figure 12. The dotted components and flows are out of the scope for risk analysis but necessary parts of the V2X application. A trust boundary is defined for an autonomous vehicle's components, where received messages are processed internally. The V2X messages come from RSUs or external vehicles as GeoNetworking packets. The messages from RSUs include SPATEM, MAPEM, CAM and DENM, while the messages from vehicles include CAM and DENM. The received V2X messages are passed to the coordination unit, which is responsible for coordinating information included in V2X messages with recognized objects by sensors and cameras. The decision unit determines the next motion of the ego-vehicle based on recognized objects, the position of the ego-vehicle itself, and warnings/notifications extracted from V2X messages. Finally, based on the plan, a signal is sent to the actuator through CAN.

The final step of stage 3 is security functional analysis and the use of trust boundaries. This step aims to analyze the security for the use cases from the functional perspective by looking at the enforcement of the security controls such as authentication, authorization and input validation through trust boundaries. In the case of the V2X application, the trust boundary is only the one that divides the internal side of a vehicle and the external world. Transactions conducted over the boundary are sending and receiving V2X messages for all

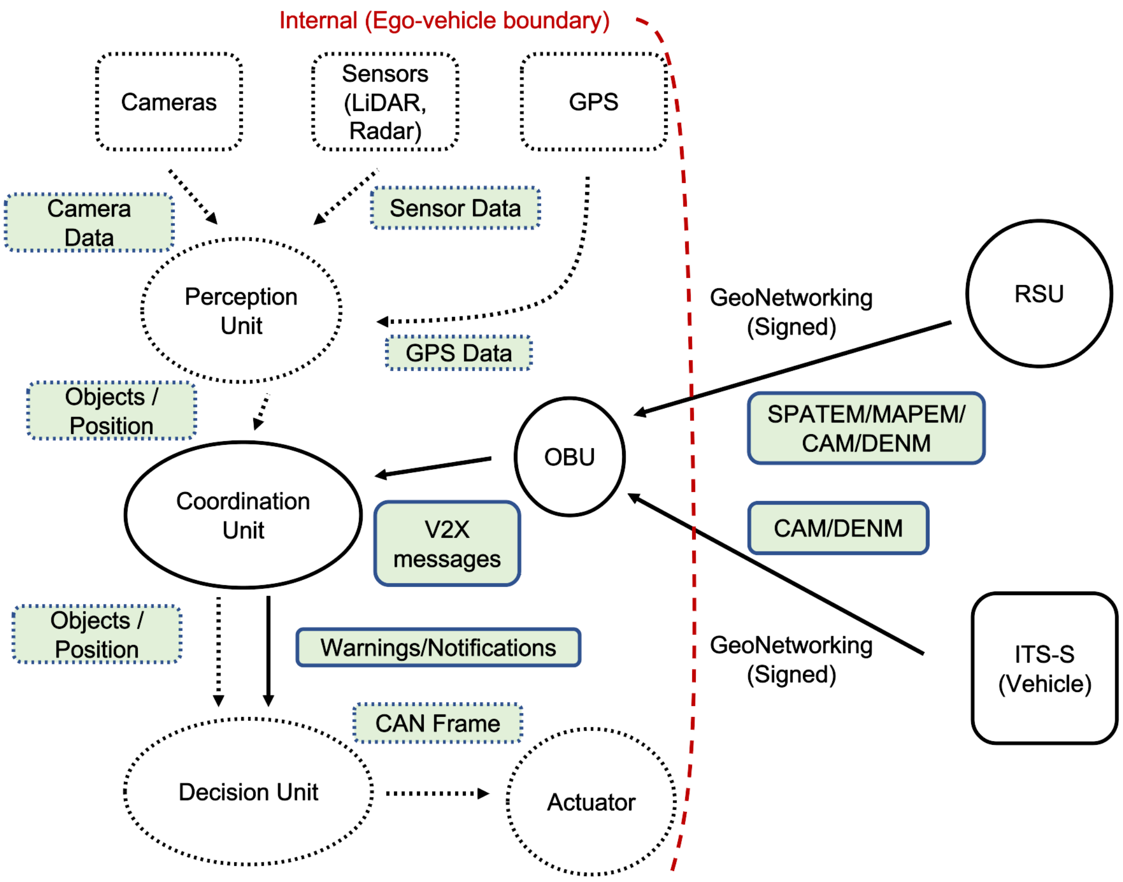


Figure 12. *Data Flow Diagram.*

| | |
|---|---|
| Threat actor | A malicious vehicle/person. |
| Motivation | Sabotage the safety of the V2X communication system. |
| Goal | Cause misbehavior of autonomous vehicles. |
| Targets | Vehicles running on the road. |

Table 5. *Threat scenario (misbehavior).*

| | |
|---|---|
| Threat actor | A malicious vehicle/person. |
| Motivation | Sabotage the safety of the V2X communication system. |
| Goal | Prevent an autonomous vehicle from receiving V2X messages. |
| Targets | Vehicles running on the road. |

Table 6. *Threat scenario (Denial of Service).*

of the use cases. Security functions involved in the transactions are verification of V2X messages based on the authorization tickets (certificates). The verification process of V2X messages and management of certificates in the verification process are analyzed, depicting flow charts and a sequence diagram in Section 3.4 and Section 3.5, respectively.

## 4.5 Threat analysis

The goal of this stage is the analysis of the threats based on various sources of threat intelligence. As the first step, threat actors and their motivation, targets and goals are correlated and summarized as threat scenarios. Threat scenarios are shown in Table 5 and Table 6. Note that these threat scenarios are not associated with specific threats. The threats are defined and associated with the threat scenarios in the following sections.

The threat actor is a malicious vehicle/person in both threat scenarios. The motivation is also the same between the two scenarios: to sabotage the safety of the V2X communication system. However, the goals are different. In Table 5, the goal of the actor is to cause the misbehavior of an autonomous vehicle. The misbehavior of an autonomous vehicle could trigger a critical accident. Therefore, the motivation of the actor, sabotage of the safety made by V2X communication, would be satisfied by achieving the goal. On the other hand, in table 6, the goal of the actor is to prevent an autonomous vehicle from receiving V2X messages. If an autonomous vehicle cannot receive V2X messages, the vehicle cannot make decisions based on the V2X messages. Even in that case, the vehicle still can be operated by perception and decision based on data from its sensor. However, disabling perception and decisions based on V2X messages could compromise safety.

The next step is to gather threat information from various sources. Typically, threat information is gathered from internal sources as well as external sources. However, available internal sources could not be found because currently V2X applications which

| Threat | Sources |
|---|---|
| Notification of false information | [60] |
| Tampering | [61] |
| DoS | [60], [61], [62] |
| Map database poisoning | [60] |
| Impersonation | [61] |
| Deception | [61] |

Table 7. *Enumerated Threats.*

follows ETSI ITS-G5 are still not operated in many vehicles including the autonomous vehicle being developed in Tallinn University of Technology. For that reason, threat analysis is conducted using the information gathered from only external sources.

Table 7 shows enumerated threats found in external sources [60], [61] and [62]. Notification of false information could cause vehicles' misbehavior. The false information includes all wrong information for the use cases shown in Table 4. Tampering means modifying the content of a V2X message in transit by active eavesdropping. DoS means denial of service in a broad meaning. Primarily, it prevents an autonomous vehicle from receiving new V2X messages. The map database poisoning targets LDM, where received messages, especially static information such as the information included in MAPEM, are temporarily stored. Impersonation is caused by a threat actor who sends messages with the identity of another vehicle. Impersonation can be caused by a Sybil attack, where a threat actor holds multiple identities. Deception is similar to impersonation. However, in deception, a threat actor pretends to be another vehicle by injecting fake information into the message. For example, a malicious vehicle could pretend to be an emergency vehicle by indicating it in the message. As another example, a malicious vehicle could include another vehicle's position as its own position and pretend to be the vehicle.

Deception and impersonation cannot be direct threats to achieve the goals: to cause misbehavior of autonomous vehicles or to prevent an autonomous vehicle from receiving V2X messages. However, both the deception and impersonation can be used to hide attacker's identity and position in other threats. Hereinafter, deception and impersonation are understood as part of other threats, namely notification of false information, map database poisoning, tampering and DoS.

The final step of this stage is a correlation of threats to assets. The system assets defined in Section 4.2 are vehicles and exchanged V2X messages. LDMs (map databases) have not been explicitly defined as a system asset, but they are included in "vehicles" because LDM is a component consisting of a vehicle to store V2X messages temporarily. To create threat model, threats in Table 7 are assigned to the DFD shown in Figure 12 as conducted in [59].

The threat model is shown in Figure 13. For simplicity, components in the ego-vehicle boundary are represented as "ego-vehicle,", where the system asset "vehicles" are depicted as "ego-vehicle" because the data flow diagram was created from the perspective of the vehicle which receives V2X messages. This representation does not affect threat analysis because the flow of data between units in a vehicle is out of scope focused in this thesis.

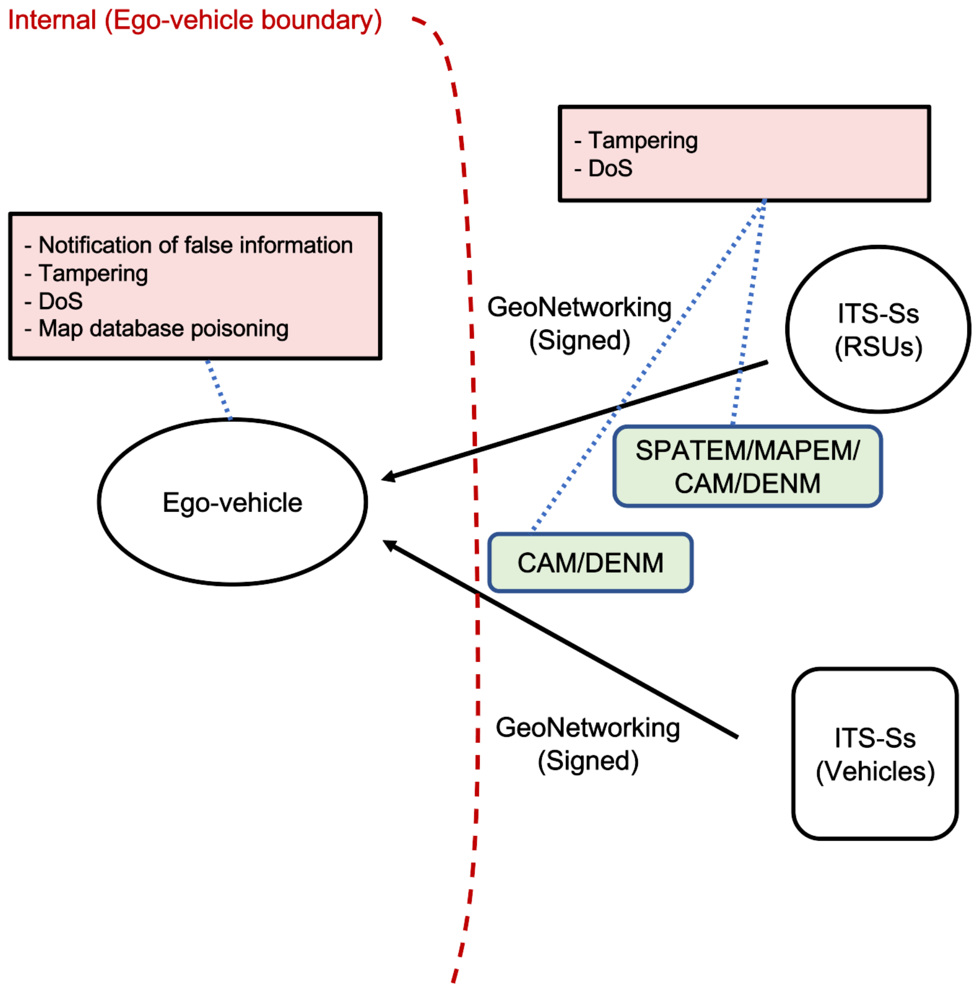


Figure 13. *Threat model.*

Notification of false information, tampering, DoS and map database poisoning are mapped to ego-vehicle. Notification of false information, tampering, map database poisoning, could cause vehicles to make wrong perceptions and decisions based on false information.DoS does not cause wrong decisions directly. However, DoS could prevent vehicles from receiving V2X messages and also cause wrong decisions. Tampering compromises the integrity of exchanged V2X messages, namely SPATEM, MAPEM, CAM and DENM. DoS could also affect exchanged V2X messages by compromising their availability.

## 4.6 Weakness and vulnerability analysis

In this stage, the weaknesses and vulnerabilities of the V2X application are analyzed. Besides, the weaknesses and vulnerabilities are correlated to the threats enumerated in the previous section. The first step in this stage is to identify weak design patterns in the architecture scope defined in Section 4.3 and decomposed in Section 4.4.

First of all, all of the ITS services focused on this thesis, namely CA, DEN, RLT and TLM are required to sign the communicated message by an authorization ticket [44, 45, 46]. On the other hand, encryption is not required. This means that threat actors can see the information included in V2X messages without participating in a trusted network supported by PKI. V2X messages do not include confidential information which can be used to identify an ITS-S uniquely. Nonetheless, pseudonyms that work as a temporal identity are still included. While pseudonyms are not changed, two different messages can be linked together. Because of the unlinkability provided by the trust and security management system of the ETSI ITS-G5 V2X communication system described in Section 3.1, it is recommended to change the pseudonyms in a fast cycle. Nonetheless, a malicious vehicle/person can obtain pseudonyms and impersonate another ITS-S while the pseudonyms of the ITS-S are not changed. Besides, a threat actor can obtain the authorization ticket of another ITS-S, though it is meaningless without holding a private key corresponding to the public key in the authorization ticket. When a private key is obtained, the threat actor can create a legitimate authorization ticket by replacing the public key in the authorization ticket with the key generated from the obtained private key. This vulnerability is summarized as *lack of packet encryption*.

By virtue of the security services of the ETSI ITS-G5 communication system supported by PKI, only authenticated and authorized ITS-Ss can participate in the V2X communication. Allowing only legitimate ITS-Ss, for example, vehicles manufactured by a trusted company (identifiable by canonical ID), makes it difficult for malicious vehicles/persons to join in the V2X communication. In such networks, where participants are strictly restricted, receivers of V2X messages do not have to take care of the information included in the V2X messages so much. Even if a message comes from a malicious vehicle/person who has not participated in the network, the invalidity of the message can be detected in the message verification process shown as Figure 8.

However, even in the trusted networks, a malicious car/person may exist. For example, suppose the requirement to participate in the trusted network is possession of a car manufactured by a trusted company. In that case, a malicious car/person can participate in the trusted network by purchasing such a car. Assuming that the private key are not accessible even from the car owner, it is difficult to generate an arbitrary V2X message that other ITS-Ss would verify. However, even in that case, if the ROM of OBU is not encrypted, it may be possible to obtain the private key by reading ROM data. Once they obtain the private key, they can generate any messages with any device and program. To summarize, *weak management of private key* can be considered a vulnerability.

As mentioned in the explanation of the vulnerability *weak management of private key*, any malicious vehicle/person can generate and send V2X messages only with a program to generate V2X messages and a device that is able to transmit messages on IEEE802.11p channel. The device can be purchased on the Internet, and the program to generate V2X messages is available as OSS.

From the perspective of the receiver side, a vulnerability is derived from weak requirements for receiving unauthorized V2X messages. The focused V2X messages, namely CAM, DENM, SPAT and MAP, are required to be signed as defined in [44], [45] and [46]. However, the receiver is not required to receive only signed messages. The statement "messages shall be signed" found in [44] and [46] could be interpreted that the communicated message must be signed and the receiver should drop unauthorized messages. However, the statement also can be interpreted that the receiver is not required to drop unauthorized messages though the sender shall sign every message. Depending on the implementation, the receiver can receive unauthorized messages as well as signed messages. For example, Vanetza [55], which is one of the most notable implementations of the ETSI ITS-G5 protocol stack, can accept unsigned messages. This vulnerability is summarized as *weak requirement for unauthorized messages*.

When diving into the technical specification of GeoNetworking Protocol, one of the forwarding schemes, GBC, allows a malicious vehicle/person to send a malformed message to multiple ITS-Ss in an arbitrary area. By GBC, the sender of a V2X message specifies a destination area, and the message is delivered to all ITS-Ss in the destination area. This means that the malicious vehicle/person could cause threats on vehicles in an arbitrary area by setting the area to the destination area. The malicious vehicle/person doesn't have to move to send messages to various areas. What the malicious vehicle/person has to do to send malformed messages to various areas is only to change the destination area in packets. This vulnerability is summarized as *arbitrary destination area setting in GBC*.

Multi-hop forwarding enabled by TSB and GBC also allows a malicious vehicle/person to send a malformed message to multiple ITS-Ss. In TSB and GBC, a sender of a V2X message can specify a maximum hop limit. Then, the V2X message is forwarded to ITS-Ss reachable by the number of hops specified as the maximum hop limit. This means that a malicious vehicle/person can deliver a malformed message to ITS-Ss, even if they are far from the malicious vehicle/person, by specifying a significant number to the maximum hop limit. This vulnerability is called *arbitrary maximum hop limit setting in TSB and GBC*.

In the GBC and TSB, an ITS-S could receive the same message multiple times. Figure 14 shows such a case, where a vehicle sends a V2X message with maximum hop limit 4. The message is forwarded three times and delivered to ITS-Ss within 4 hops. As shown in Figure 14, paths of 4 hops could include a closed-loop. This means that an ITS-S receives the same message multiple times. When an ITS-S receives a V2X message that the ITS-S has already received, a duplicate packet detection algorithm works, and the V2X message is dropped [63]. However, even in that case, the packet is once captured, and the duplicate packet detection algorithm works. The duplicate packet detection algorithm is linear algorithm [63]. Therefore, it is not so time-consuming. However, if many messages are received, the ITS-S has to detect duplicated messages repeatedly.

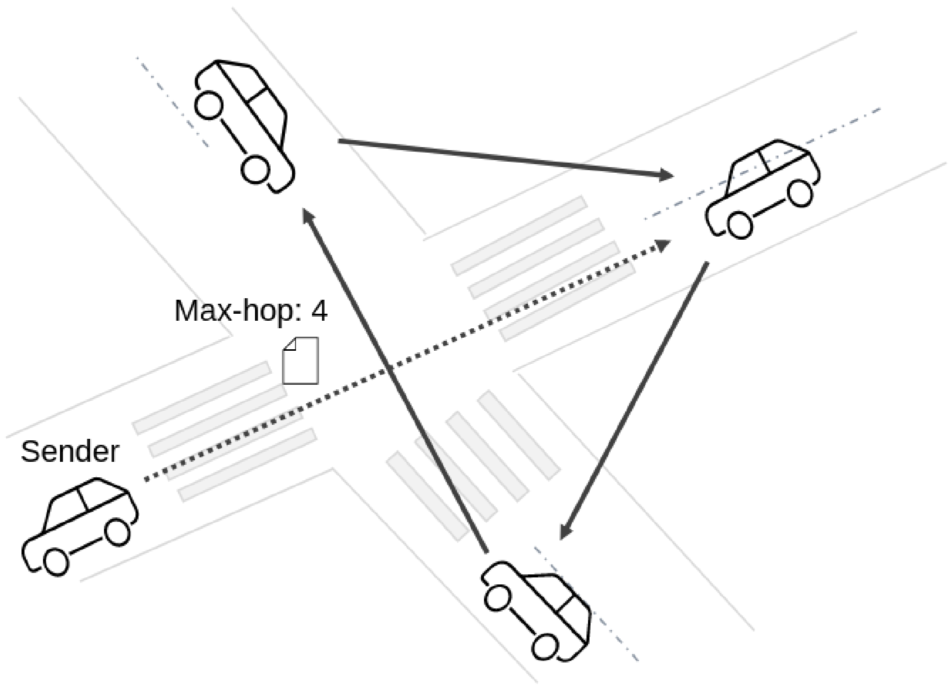


Figure 14. *Closed loop in TSB and GBC.*

Next, a vulnerability underlying the message verification process is described. As shown in Figure 9, When an ITS-S receives an authorized message, the ITS-S has to check the validity of the sender's certificate as well as the validity of AA's certificate. Besides, as shown in Figure 8, the authenticity and integrity of a V2X message have to be verified using the public key and signature. Suppose a malicious vehicle/person forces other ITS-Ss

| ID | Vulnerability |
|---|---|
| 1 | lack of packet encryption |
| 2 | Weak management of private key |
| 3 | Weak requirement for unauthorized messages |
| 4 | Arbitrary destination area setting in GBC |
| 5 | Arbitrary maximum hop limit setting in TSB and GBC |
| 6 | Possibility to send enormous messages with false signature |

Table 8. *Identified vulnerabilities.*

| Asset | Threat | Vulnerabilities |
|---|---|---|
| Vehicle | Notification of false information | 1,2,3,4,5 |
| | Tampering | 1,2 |
| | DoS | 4,5 |
| | Map database poisoning | 1,2,3,4,5 |
| Exchanged messages | Tampering | 1,2 |
| | DoS | 1,2,4,5 |

Table 9. *Mapping of identified vulnerabilities to threats.*

to conduct the verification process repeatedly by sending V2X messages with a false signature or false hash of the issuer enormous times. If the time to complete verification is larger than the frequency of sending messages, the computational resource would be flooded. Here, the malicious vehicle/person doesn't have to obtain a private key. What the threat actor has to do is only to add a false signature to a message. This vulnerability is summarized as *possibility to send enormous messages with false signature*.

The identified vulnerabilities are summarized in Table 8. The assigned IDs are used in the next step.

The next step of this stage is mapping the identified vulnerabilities to threats described in Section 4.5. Table 9 shows mapping of the vulnerabilities (in Table 8) to threats (in Figure 13). The asset represents one of the assets to which threats are mapped in Figure 13. The vulnerabilities are some of the IDs of vulnerabilities in Table 8. How the vulnerabilities are exploited to cause each threat is described in the next section.

## 4.7 Attack modeling

The goal of this stage is to analyze and model the attacks against the V2X application. The first step of this step is the enumeration of attack scenarios. The attack scenarios are created for each threat. The attacker is always a malicious vehicle/person as described in threat scenarios shown in Table 5 and Table 6.

### 4.7.1 Attacker's capabilities

The existence of an attacker with some resources and capabilities are assumed to consider attack scenarios. First, it is assumed that the attacker does not have an authorization ticket to send authorized messages. However, the attacker has the following resources.

- (a car) a purchased old car which has an OBU. The ROM of the OBU is not encrypted.
- (an OBU) an OBU purchased on the Internet, to which the malicious vehicle/person can login by SSH and dump received messages using a tool such as tcpdump and wireshark.
- (a program) a publicly available program to generate V2X messages such as Vanetza [55].

The attacker tries to utilize the OBU purchased on the Internet and the program to cause threats to other vehicles. The OBU in the purchased old car is not used to capture and transmit messages, but the private key of the OBU in the car is sometimes extracted. The attacker's capabilities with the resources are assumed as listed below:

- The attacker can obtain the private key from the not-encrypted OBU in the purchased old car.
- The attacker can generate an arbitrary GeoNetworking packet with the program to generate V2X messages. The attacker can set any forwarding schemes, any destination area, and any maximum hop limit to the GeoNetworking packet header.
- The attacker can generate an arbitrary V2X message with the program to generate V2X messages. The attacker can put any wrong information in the V2X message.
- The attacker can generate a signature from the generated GeoNetworking packet and V2X message payload with the obtained private key.
- The attacker can transmit the GeoNetworking packet with the V2X message's payload using the OBU purchased on the Internet.
- The attacker can capture V2X messages in transit by the OBU purchased on the Internet. The OBU always listens to the IEEE 802.11p channel. Therefore, all of the packets on the IEEE802.11p channel can be captured by the attacker.

### 4.7.2 Attack Scenarios

The Attack Scenarios (AS) by the attacker with the capabilities defined in the previous section against the V2X application are listed below.

**AS1 (Notification of false information)**

1. Obtain the authorization ticket of an ITS-S from received messages at the OBU the vulnerability 1.
2. Obtain the private key from the OBU in the purchased car exploiting the vulnerability 2.
3. Generate the public key corresponding to the obtained private key.
4. Replace the public key in the obtained authorization ticket with the generated public key.
5. Determine a target area.
6. Generate a skeleton of GeoNetworking packet.
7. Set the target area to the destination area.
8. Set a large number to the maximum hop limit to reach the destination area exploiting vulnerability 4 and vulnerability 5.
9. Set GBC to the forwarding scheme of GeoNetworking.
10. Generate a V2X message payload including false information such as the existence of a fake vehicle relevant to the targeted area.
11. Generate signature with the obtained private key.
12. Construct a secured GeoNetworking packet using the authorization ticket, signature and the message payload.
13. Transmit the secured GeoNetworking packet.
14. A vehicle which received the V2X message verifies the signature and accepts the message.
15. The vehicle uses information included in the accepted message for controlling the vehicles.

**AS2 (Notification of False information)**

1. Generate a skeleton of GeoNetworking packet.
2. Set SHB to the forwarding scheme of GeoNetworking.
3. Generate a V2X message payload including false information such as the existence of a fake vehicle relevant to the targeted area.
4. Construct a GeoNetworking packet using the message payload.
5. Transmit the GeoNetworking packet.
6. A vehicle which does not drop unauthorized messages accepts the message, where vulnerability 3 is exploited.
7. The vehicle uses information included in the accepted message for controlling the vehicle.

**AS3 (Tampering)**

1. Obtain the authorization ticket of an ITS-S from received messages at the OBU the

vulnerability 1.

2. Obtain the private key from the OBU in the purchased car exploiting the vulnerability 2.
3. Generate the public key corresponding to the obtained private key.
4. Replace the public key in the obtained authorization ticket with the generated public key.
5. Capture a V2X message at the OBU.
6. Tamper the received V2X message payload and include wrong information.
7. Generate signature with the obtained private key.
8. Construct a secured GeoNetworking packet using the authorization ticket, signature and the tampered message payload.
9. Retransmit the secured GeoNetworking packet to the destination specified in the received packet.
10. A vehicle which received the V2X message verifies the signature and accepts the message.
11. The vehicle uses information included in the accepted message for controlling the vehicle.

**AS4 (Map database poisoning)**

1. Obtain the authorization ticket of an ITS-S from received messages at the OBU the vulnerability 1.
2. Obtain the private key from the OBU in the purchased car exploiting the vulnerability 2.
3. Generate the public key corresponding to the obtained private key.
4. Replace the public key in the obtained authorization ticket with the generated public key.
5. Determine a target area.
6. Generate a skeleton of GeoNetworking packet.
7. Set the target area to the destination area.
8. Set a large number to the maximum hop limit to reach the destination area exploiting vulnerability 4 and vulnerability 5.
9. Set GBC to the forwarding scheme of GeoNetworking.
10. Generate a V2X message payload including false information such as the wrong lane topology and allowed maneuver relevant to the targeted area.
11. Generate signature with the obtained private key.
12. Construct a secured GeoNetworking packet using the authorization ticket, signature and the message payload.
13. Transmit the secured GeoNetworking packet.
14. A vehicle which received the V2X message verifies the signature and accepts the

message.

15. The vehicle stores the false information to LDM (map database).
16. The vehicle uses information included in the accepted message for the core process of self-driving.

**AS5 (Map database poisoning)**

1. Generate a skeleton of GeoNetworking packet.
2. Set SHB to the forwarding scheme of GeoNetworking.
3. Generate a V2X message payload including false information such as the wrong lane topology and allowed maneuver relevant to the targeted area.
4. Construct a GeoNetworking packet using the message payload.
5. Transmit the GeoNetworking packet.
6. A vehicle which does not drop unauthorized messages accepts the message, where vulnerability 3 is exploited.
7. The vehicle stores the information to LDM (map database).
8. The vehicle uses information included in the accepted message for controlling the vehicle.

**AS6 (DoS)**

1. Obtain the authorization ticket of an ITS-S from received messages at the OBU the vulnerability 1.
2. Obtain the private key from the OBU in the purchased car exploiting the vulnerability 2.
3. Generate the public key corresponding to the obtained private key.
4. Replace the public key in the obtained authorization ticket with the generated public key.
5. Generate a skeleton of GeoNetworking packet.
6. Set a large number to the maximum hop limit vulnerability 5.
7. Set TSB to the forwarding scheme of GeoNetworking.
8. Generate a V2X message payload.
9. Generate signature with the obtained private key.
10. Construct a secured GeoNetworking packet using the authorization ticket, signature and the message payload.
11. Transmit the secured GeoNetworking packet repeatedly.
12. A vehicle which received the V2X message verifies the signature.
13. The vehicle detects duplicate packets by duplicate packet detection.
14. Repeat 1–13.
15. The computational resource of the vehicle is flooded.

**AS7 (DoS)**

1. Generate a skeleton of GeoNetworking packet.
2. Set SHB to the forwarding scheme of GeoNetworking.
3. Generate a V2X message payload.
4. Generate a false signature.
5. Construct a GeoNetworking packet using the message payload and the false signature.
6. Transmit the GeoNetworking packet.
7. A vehicle which received the V2X message drop the packet in the verification process.
8. Repeat 1–7.
9. The computational resource of the vehicle is flooded.

AS1, AS3 and AS4 include obtaining an authorization ticket of another ITS-S and the private key from the OBU in the purchased car. After obtaining the private key and the authorization ticket exploiting the vulnerability *lack of packet encryption* and *weak management of private key*, respectively, threat actors can transmit authorized messages toward any areas exploiting the vulnerability *arbitrary destination area setting in GBC*. Besides, by setting a large number to the maximum hop limit, the messages can be delivered to the area exploiting vulnerability *arbitrary maximum hop limit setting in TSB and GBC*. Therefore, a threat actor can set a small area and send malformed messages, including information relevant to the area, even if the area is far from the threat actor. Figure 15 shows an example of a false event notification to a vehicle in the destination area far from the threat actor. First, a threat actor determines a target area. Next, the threat actor generates a GeoNetworking packet, where the destination area is set to the target area. The message included in the packet has information about a fake vehicle's existence in the destination area. Finally, the threat actor sets a large maximum hop limit to make make the message delivered to the destination area. Then, the message is delivered to the destination area by multi-hop forwarding.

On the other hand, in AS2 and AS5, a threat actor sends an unauthorized message to ITS-Ss. In this thesis, let me assume that the number of ITS-Ss which accepts and forward unauthorized messages is so limited that unauthorized messages cannot be delivered to ITS-Ss far from the threat actor. On the assumption, setting a significant number to the maximum hop limit to make a message delivered to an area far from the threat actor is not an effective way. However, even without multi-hop forwarding, the threat actor can make unauthorized messages accepted by nearby ITS-Ss, setting SHB to the forwarding scheme of GeoNetworking, while exploiting the vulnerability *weak requirement for unauthorized messages*. Vehicles reachable by single-hop are the ones in the vicinity of the threat actor. To cause false information notification (AS2) and map database poisoning (AS5) to the

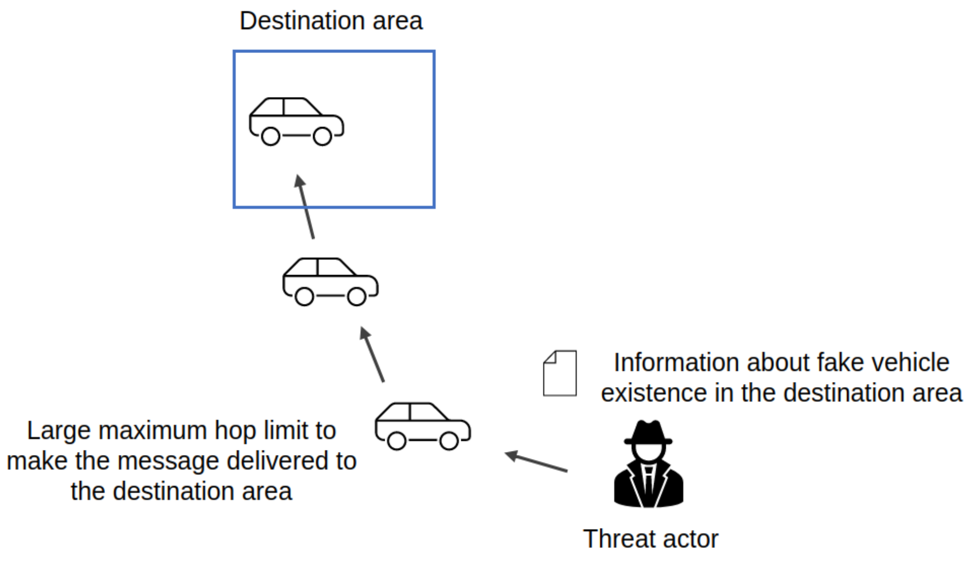

Destination area
Information about fake vehicle existence in the destination area
Large maximum hop limit to make the message delivered to the destination area
Threat actor

Figure 15. *False event notification to a vehicle in the destination area far from the threat actor.*

vehicles in the vicinity, the information included in the message should also be the wrong information about the traffic environment in the vicinity.

In AS6, a threat actor sends enormous messages with a large maximum hop limit exploiting *arbitrary maximum hop limit setting in TSB and GBC* after obtaining the private key and the authorization ticket exploiting the vulnerability *lack of packet encryption* and *weak management of private key*, respectively. Then, the threat actor makes vehicles do the duplicate packet detection enormous times and consume computational resources. Hereinafter, the attack vector to flood computational resources by sending enormous GBC/TSB packets with a large hop limit is called "TSB/GBC flood." On the other hand, in AS7, a threat actor sends enormous messages with a false signature. Then, the receiver is forced to verify the message with a false signature repeatedly, which could exhaust computational resources. Hereinafter, the attack vector to flood computational resources by sending enormous packets with a false signature is called "False signature flood."

The next step is an enumeration of attack vectors. In this thesis, attack vectors are defined as follows: critical steps in attack scenarios that determine whether the attack can be performed or not. The attack vectors are enumerated based on the attack scenarios. The Attack Vectors (AV) and the corresponding threats (attack scenarios) are shown in Table 10.

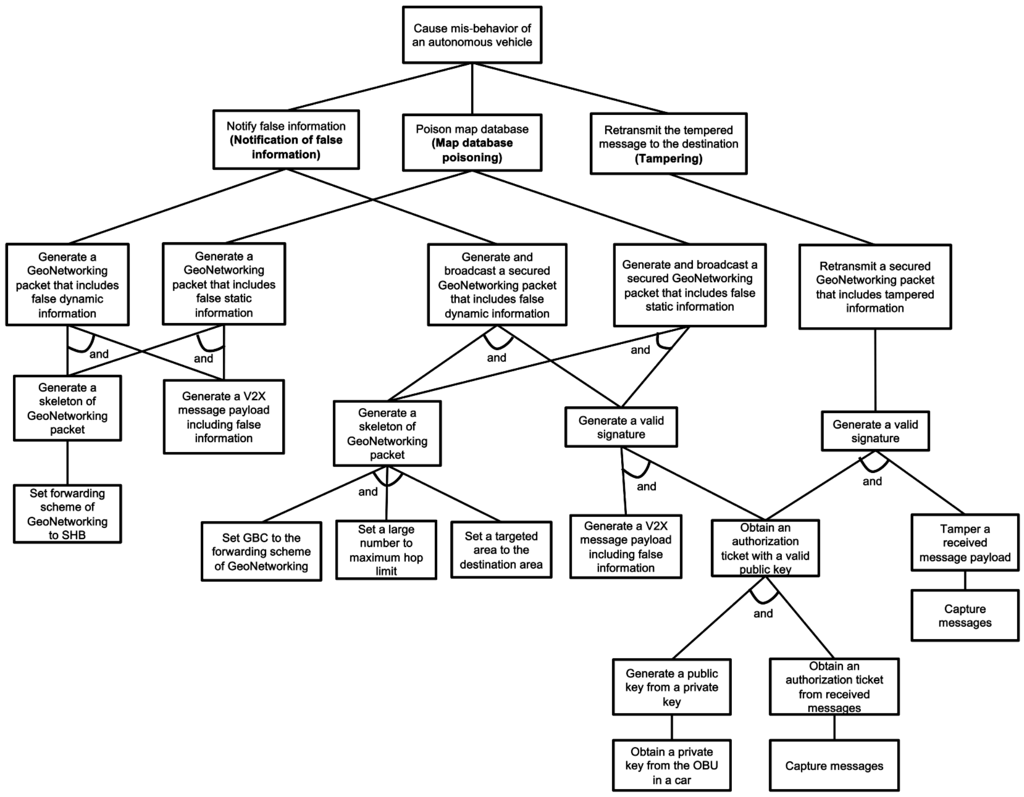
Next, attack trees are created to assess the probability of each attack scenario. In the attack trees, attacks against a system are represented in a tree structure, with the goal as the root node and steps to achieve that goal as leaf nodes [64]. The goals are chosen

| ID | Attack Vector | Attack scenarios |
|---|---|---|
| AV1 | TSB/GBC flood: send a V2X message with a large maximum hop limit. | DoS (AS6) |
| AV2 | False signature flood: send a V2X message with a false signature many times. | DoS (AS7) |
| AV3 | Obtaining a private key and creating an valid authorization ticket | Notification of false information (AS1), Tampering (AS3), Map database poisoning (AS4), DoS (AS6) |
| AV4 | Single-Hop Broadcast (SHB) of a message to ITS-Ss so that an unauthorized message is accepted | Notification of false information (AS2), Map database poisoning (AS5) |
| AV5 | GeoBroadcast (GBC) of a message to an targeted area with a large maximum hop limit. | Notification of false information (AS1), Map database poisoning (AS4) |
| AV6 | Retransmission of a message to ITS-Ss which have been set as the destinations after capturing the message and modifying the content | Tampering (AS3) |

Table 10. *Attack vectors.*

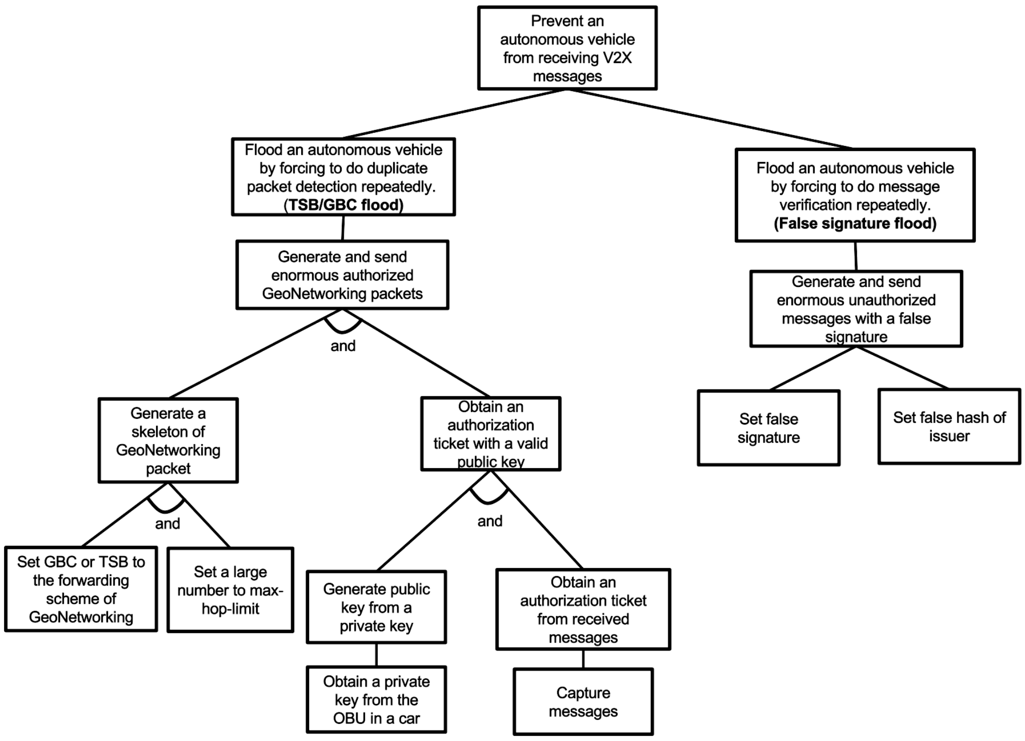
from threat scenarios shown in Table 5 and Figure 6. The nodes except for the root node are created based on the attack scenarios from AS1 – AS9, especially focusing on attack vectors (in 10). An attack tree with the goal *cause misbehavior of an autonomous vehicle* is shown in Figure 16. Note that it is assumed the targeted vehicle uses the received message for controlling the vehicle. On the other hand, an attack tree with the goal *prevent an autonomous vehicle from receiving V2X messages* is shown in Figure 17.

From Figure 16, notification of false information, map database poisoning and tampering can be threats to achieve the goal: cause misbehavior of an autonomous vehicle. Notification of false information, map database poisoning has two attack chains. One attack chain involves obtaining a private key and creating a valid authorization ticket. The most difficult task in this attack chain is to obtain a private key because it requires reading the private key from the OBU in a car. Technical skills to extract a private key from the ROM are necessary. Therefore, it can be said that the attack scenarios involving obtaining a private key and creating a valid authorization ticket are comparatively difficult to achieve. However, once a private key is obtained, the threat actor can create a valid authorization ticket with the public key and a valid signature generated from the obtained private key. Then, the threat actor can broadcast an authorized GeoNetworking packet including a malformed message with GBC, where the maximum hop limit is set to a large number. The other attack chain requires threat actors to make the malformed messages accepted. This means that there must be a vehicle that has the vulnerability *weak requirement for unauthorized messages* in the range reachable by a single hop. In this thesis, it has been assumed that the number of ITS-Ss which accepts and forward unauthorized messages is



Cause mis-behavior of an autonomous vehicle
Notify false information **(Notification of false information)**
Poison map database **(Map database poisoning)**
Retransmit the tempered message to the destination **(Tampering)**
Generate a GeoNetworking packet that includes false dynamic information
Generate a GeoNetworking packet that includes false static information
Generate and broadcast a secured GeoNetworking packet that includes false dynamic information
Generate and broadcast a secured GeoNetworking packet that includes false static information
Retransmit a secured GeoNetworking packet that includes tampered information
and
and
and
and
and
and
and
Generate a skeleton of GeoNetworking packet
Generate a V2X message payload including false information
Generate a skeleton of GeoNetworking packet
Generate a valid signature
Generate a valid signature
Set forwarding scheme of GeoNetworking to SHB
Set GBC to the forwarding scheme of GeoNetworking
Set a large number to maximum hop limit
Set a targeted area to the destination area
Generate a V2X message payload including false information
Obtain an authorization ticket with a valid public key
Tamper a received message payload
Capture messages
Generate a public key from a private key
Obtain an authorization ticket from received messages
Obtain a private key from the OBU in a car
Capture messages

Figure 16. *Attack tree with the goal: cause misbehavior of an autonomous vehicle.*



Prevent an autonomous vehicle from receiving V2X messages
Flood an autonomous vehicle by forcing to do duplicate packet detection repeatedly. **(TSB/GBC flood)**
Flood an autonomous vehicle by forcing to do message verification repeatedly. **(False signature flood)**
Generate and send enormous authorized GeoNetworking packets
and
Generate and send enormous unauthorized messages with a false signature
Generate a skeleton of GeoNetworking packet
Obtain an authorization ticket with a valid public key
Set false signature
Set false hash of issuer
and
and
Set GBC or TSB to the forwarding scheme of GeoNetworking
Set a large number to max-hop-limit
Generate public key from a private key
Obtain an authorization ticket from received messages
Obtain a private key from the OBU in a car
Capture messages

Figure 17. *Attack tree with the goal: prevent an autonomous vehicle vehicle from receiving V2X messages*

so limited. Therefore, the probability of performing attacks following this attack chain is not so high. For tampering, GeoBroadcasting of messages to a target area cannot be adopted because a threat actor has to retransmit the received GeoNetworking packet to ITS-Ss, which have been set as the destination.

From Figure 17, TSB/GBC flood requires threat actors to obtain the private key and create a valid authorization ticket of an ITS-S. The most difficult task in this attack chain is to obtain a private key because it requires reading the private key from the OBU in a car. Once a private key is obtained, the threat actor can create a valid authorization ticket with the public key and a valid signature generated from the obtained private key. Then, threat actors can conduct the TSB/GBC flood by sending enormous messages with a large maximum hop limit. Threat actors sometimes need multiple devices capable of transmitting data on the IEEE802.11 channel to overwhelm the victim's computational resource. To make a vehicle do the verification process repeatedly is easily performed because messages need not be accepted. The message would not be accepted in the first place because the message does not have a valid signature. What a threat actor has to do for the false signature flood is to send enormous messages which would not be accepted so that the verification process floods the computational resource.

However, the probability of causing DoS depends on the capacity of the ITS-S to process messages. If the ITS-S has a lot of computational resources, it becomes hard to accomplish a DoS attack. The spec of CPU can be estimated more or less as described in Section 4.3. Let me assume that an ITS-S sends V2X messages by $1000$ Hz with the intention to flood computational resources. Under that condition, the number of messages delivered in one second is over $1000$ messages including messages from other ITS-Ss. If the targeted vehicle cannot perform duplicate packet detection or verification of the message in $\frac{1}{1000}$ s $=$ $1$ ms, it can be said that the computational resource would be flooded. From this estimation, it can be said that performing a DoS attack is doable. Detailed experiments are conducted in Section 8.

## 4.8 Risk analysis

The first step of this stage is the calculation of the risk value for each threat analyzed in the previous sections. Risk value $R$ is calculated by the following equation:

$$R = T \times I \tag{4.1}$$

, where $T$ is the probability that the threat is caused by exploiting one or more vulnerabilities, while $I$ is the scale of the impact made by the threat [59]. Both threat probabilities and impact are rated by five levels. Threat probabilities are rated by VERY LOW (rare), LOW (unlikely), MEDIUM (possible), HIGH (likely) or VERY HIGH (very likely). Impacts are rated in the context of safety in the V2X communication system as described in Section 4.2. The rates are VERY LOW (negligible), LOW (could cause some disturbance), MEDIUM (could cause some damages), HIGH (could cause a critical accident) and VERY HIGH (could cause a highly critical accident). The assessment metrics are shown in Table 11 with more details.

In the discussion about the attack tree shown in Figure 16, it is deduced that generating an authorized message with the private key and a valid authorization ticket is a difficult task. Besides, making an unauthorized message accepted by the vehicles reachable by one-hop is also a difficult task due to the limited number of vehicles which has the vulnerability *weak requirement for unauthorized messages*. However, these tasks are still possible depending on the skill of a threat actor and the existence of a vulnerable vehicle. For these reasons, the probability of notification of false information, map database poisoning, which requires obtaining the private key and a valid authorization ticket, or making an authorized message accepted, is considered MEDIUM. Tampering also requires obtaining the private key and a valid authorization ticket. Therefore, the probability of tampering is also considered MEDIUM. Notification of false information, map database poisoning and tampering impact vehicles and could result in misbehavior of the vehicles (loss of safety of vehicles) by modifying the included information to a fake information. The impact of tampering can be said HIGH because misbehavior of vehicles could result in critical accidents. The impact of false event notification and map database poisoning is considered VERY HIGH because an threat actor can cause threats to a lot of vehicles by iterating over a large region while changing the destination area in the region.

From the discussion about the attack tree shown in Figure 17, TSB/GBC flood is hard to achieve because threat actors need to obtain the private key and create a valid authorization ticket. The probability of causing TSB/GBC flood-based DoS is considered the same as the probability of tampering (MEDIUM) because the attack chain is similar. False signature flood-based DoS are easy to perform. The false signature flood attack does not require threat actors to make receivers accept messages. Besides, it is doable from the perspective of the computational resources of OBU. For these reasons, the probability of false signature flood-based DoS is considered VERY HIGH. DoS results in loss of availability of exchanged messages, which also leads to loss of safety of vehicles. However, the unavailability of messages does not cause misbehavior directly. The compromised vehicle by DoS can still operate with limited capability, where messages coming from the external world cannot be used. Therefore, the impact is considered LOW.

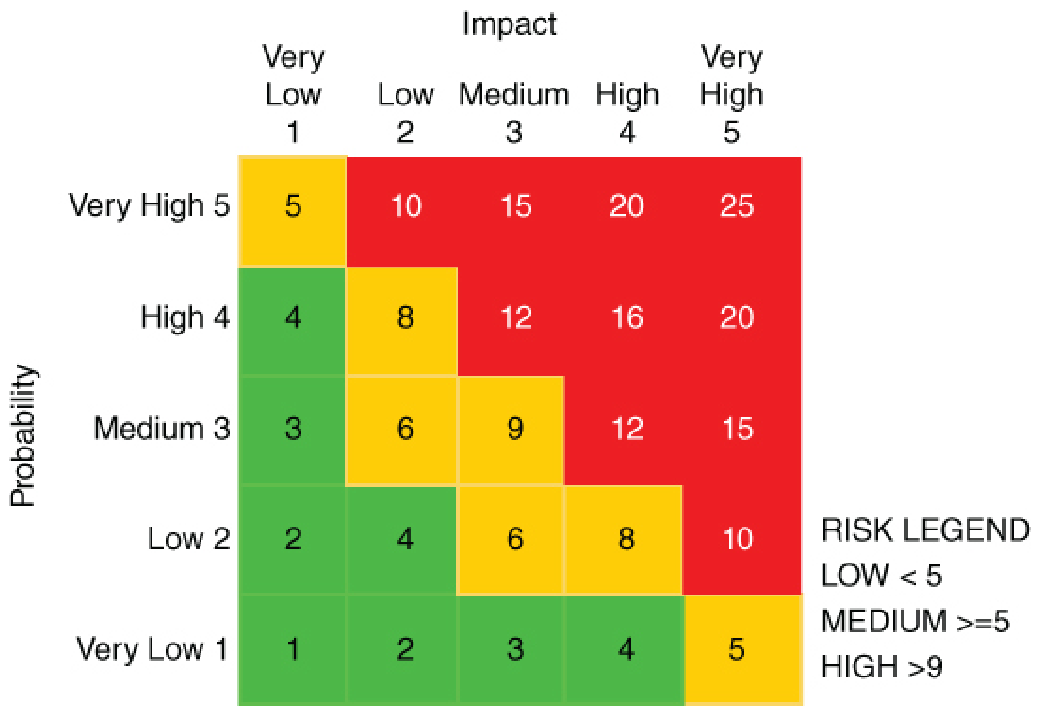
| **Threat/Impact** | **Rate** | **description** |
|---|---|---|
| Threat probability | VERY LOW | To cause the threat is almost impossible considering the difficulty of performing attacks. |
| | LOW | It is difficult to cause the threat, but possible with expensive resources and a sophisticated understanding of V2X communication and the device used for the communication. |
| | MEDIUM | It is possible to cause the threat with resources and capabilities shown in the beginning of Section 4.7. It is required for an attacker to understand the technical specification of the ETSI ITS-G5 V2X communication system. |
| | HIGH | It is possible to cause the threat with only several resources and capabilities shown in the beginning of Section 4.7. But, it is required for an attacker to understand the technical specification of the ETSI ITS-G5 V2X communication system. |
| | VERY HIGH | It is possible to cause the threat with only several resources and capabilities shown in the beginning of Section 4.7. Also, it is not required for an attacker to understand the technical specification of the ETSI ITS-G5 V2X communication system. |
| Impact | VERY LOW | The impact could make the ego-vehicle to change movement from the optimized path plan computed by autonomous vehicle core process. The movement change does not affect other vehicles' behavior. |
| | LOW | The impact could make the ego-vehicle to change movement from the optimized path plan computed by the autonomous vehicle core process. The movement change could also affect other vehicles' behavior, but does not cause any accidents. |
| | MEDIUM | Accidents that injure the ego-vehicle (*e.g.* get scrached) could be caused. |
| | HIGH | Accidents that harm the ego-vehicle (*e.g.* get dented) and make the driver injured could be caused. The accident could involve another vehicle and a pedestrian. |
| | VERY HIGH | Accidents that destroy the vehicle or kill the driver-could be caused. The accident could involve several other vehicles and pedestrians. |

Table 11. *Risk assessment metrics.*

| **Threat** | **Probability** | **Impact** | **Risk value** |
|---|---|---|---|
| Notification of false information | MEDIUM (3) | HIGH (5) | 15 (HIGH) |
| Map database poisoning | MEDIUM (3) | VERY HIGH (5) | 15 (HIGH) |
| Tampering | MEDIUM (3) | HIGH (4) | 12 (HIGH) |
| DoS (TSB/GBC flood) | MEDIUM (3) | LOW (2) | 6 (MEDIUM) |
| DoS (False signature flood) | VERY HIGH (5) | LOW (2) | 10 (HIGH) |

Table 12. *Calculated risks.*

From the above discussion, the risk for each threat is calculated as shown in Table 12 following the Equation 4.1. Besides, a label (*e.g.* LOW, MEDIUM or HIGH) is assigned based on Figure 18 (derived from [59]).



| Probability \ Impact | Very Low 1 | Low 2 | Medium 3 | High 4 | Very High 5 |
|---|---|---|---|---|---|
| Very High 5 | 5 | 10 | 15 | 20 | 25 |
| High 4 | 4 | 8 | 12 | 16 | 20 |
| Medium 3 | 3 | 6 | 9 | 12 | 15 |
| Low 2 | 2 | 4 | 6 | 8 | 10 |
| Very Low 1 | 1 | 2 | 3 | 4 | 5 |

RISK LEGEND
LOW < 5
MEDIUM >=5
HIGH >9

Figure 18. *Risk matrix (derived from [59].)*

The next step is the consideration of countermeasures based on the risk analysis. The countermeasures are considered only for the risks with HIGH. The high risk value of notification of false information and map database poisoning is partly attributed to the vulnerability *the weak requirement for unauthorized messages*. Therefore, an effective countermeasure is *installation of an additional system that blocks unauthorized messages*.

The vulnerability *arbitrary destination area setting in GBC* and the vulnerability *arbitrary* *maximum hop limit setting in TSB and GBC* can be exploited for notification of false information and map database poisoning. In general use cases of the V2X communication system, ITS-Ss do not need to set a large number to the maximum hop limit. Besides, ITS-Ss do not set a far area to the destination area. These are because messages are exchanged between ITS-Ss in the vicinity in general. For example, ITS-Ss do not need to know the state of an ITS-S 1 km away. For that reason, messages with a large maximum hop limit or messages generated far from the ego-vehicle could be malicious. Therefore, an effective countermeasure is *installation of a system that blocks GBC packets, where the maximum hop limit is large or the position of the source which generated the packet is far from the ego-vehicle*.

DoS by the false signature flood should be mitigated in a different way because the messages seem authorized, including a signature, and GBC is not used. Each ITS-S can be identified by temporal identities provided by Enrollment Authority (EA), such as GeoNetworking address. Therefore, blocking subsequent messages coming from an ITS-S in case multiple messages from the ITS-S could not be verified in a row can be an effective countermeasure. However, the threat actor can spoof the identity for every transmission of a message by impersonation. In order to mitigate the spoofing, *allowing messages coming from only trusted ITS-Ss*, which has been defined in advance, and does not perform the verification process, is an effective countermeasure.

When a malicious message is detected, the authorization ticket (certificate) embedded to the malicious message should be revoked. AA (Authorization Authority) in Figure 5 can revoke the certificate. After that, the threat actor cannot generate an authorized message which would be accepted by other ITS-Ss. However, to confirm whether the certificate included in the received message has been revoked or not, CRL (Certificate Revocation List) should be checked. In ETSI TS 102 940 V1.3.1, an optional functional element named distribution center is defined in the security management system of the ETSI ITS-G5 V2X communication [31]. The distribution center is in charge of providing ITS-Ss the updated trust information necessary for performing the verification process shown in Figure 7 and Figure 8. Even if the certificate included in the message is compatible with the certificate of AA, the certificate may be revoked. For such cases, the distribution center can be used to distribute CRL to ITS-Ss so that the ITS-Ss can verify that the included authorization ticket has not been revoked. Besides, the public key corresponding to the attacker's private key should also be revoked and notified to ITS-Ss.

In summary, effective countermeasures are as follows:

- Blocking unauthorized messages

- Blocking GBC packets with malicious configuration
- Only allowing messages coming from trusted ITS-Ss
- Deployment of a distribution center

The risks with HIGH value, AVs to cause the threats and countermeasures are summarized into Table 13 based on Table 8 (*identified vulnerabilities*), Table 10 (*attack vectors*), Table 12 (*calculated risks*) and the list of countermeasures.

In this thesis, as a combination of the countermeasures, namely blocking unauthorized messages, blocking GBC packets with malicious configuration and only allowing messages coming from trusted ITS-Ss, Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) that has these message blocking/allowing features is proposed. Whether a message is signed or not, the maximum hop limit of the message, destination area of the message, source position, and sender's identity are all included in a GeoNetworking packet. Therefore, they are detectable as patterns observed in the GeoNetworking packet. Signature-based IDS/IPS can detect patterns of packets and blocks/allows messages based on the detection results. For that reason, the IDS/IPS is developed as a signature-based one. Signature-based IDS/IPS specifically designed for the ETSI ITS-G5 V2X communication system has not been proposed nor developed in any research as described in Chapter 2. Therefore, the IDS/IPS is developed in this research. In the next chapter, data included in the GeoNetworking packet is explained in detail.

On the other hand, the deployment of a distribution center can be achieved without developing a new system. The distribution center is already defined in ETSI standards [31]. Besides, solutions that are equivalent to the distribution center have been proposed by other research work such as [28], [29] and [30].

| **Risk** | **Attack vectors** | **Countermeasures** |
|---|---|---|
| Notification of false information exploiting lack of packet encryption (vulnerability 1), weak management of private key (vulnerability 2), weak requirement for unauthorized messages (vulnerability 3), arbitrary destination area setting in GBC (vulnerability 4) and arbitrary maximum hop limit setting in TSB and GBC (vulnerability 5), which could result in loss of safety of vehicles. | Obtaining the private key and the authorization ticket of an ITS-S (AV3), Single-Hop Broadcast of a message to ITS-Ss so that an unauthorized message is accepted (AV4), GeoBroadcast of a message to an targeted area with large maximum hop limit (AV5) | Blocking unauthorized messages, Blocking GBC packets with malicious configuration |
| Map database poisoning exploiting lack of packet encryption (vulnerability 1), weak management of private key (vulnerability 2), weak requirement for unauthorized messages (vulnerability 3), arbitrary destination area setting in GBC (vulnerability 4) and arbitrary maximum hop limit setting in TSB and GBC (vulnerability 5), which could result in loss of safety of vehicles. | Obtaining the private key and the authorization ticket of an ITS-S (AV3), Single-Hop Broadcast of a message to ITS-Ss so that an unauthorized message is accepted (AV4), GeoBroadcast of a message to an targeted area with large maximum hop limit (AV5) | Blocking unauthorized messages, Blocking GBC packets with malicious configuration |
| Tampering exploiting ack of packet encryption (vulnerability 1), weak management of private key (vulnerability 2), which could result in loss of safety of vehicles. | Obtaining the private key and the authorization ticket of an ITS-S (AV3), Retransmission of a message to ITS-Ss which have been set as the destinations after capturing the message and modifying the content (AV6) | Blocking unauthorized messages, Blocking GBC packets with malicious configuration) |
| DoS exploiting possibility to send enormous messages with false signature (vulnerability 5), which could result in availability of exchanged messages and also loss of safety of vehicles. | AV2 (False signature flood: send a V2X message with false signature many times) | Only allowing messages coming from trusted ITS-Ss |

Table 13. *Mapping of risks with HIGH value, AVs to cause the threats and countermeasures.*

# 5. GeoNetworking packet structure for IDS/IPS development

In this chapter, data included in V2X message, in particular data included in GeoNetworking packet payload, are described based on [63]. The data will be used for intrusion detection and prevention to be developed.

## 5.1 GeoNetworking

### 5.1.1 Overall structure of GeoNetworking packet

In the following sections, the technical specifications of the GeoNetworking protocol are described. Some of the content included in the GeoNetworking header can be used to determine whether the received packets should be processed or not.

The structure of GeoNetworking packet is shown in Figure 19 (derived from [63]). When security is enabled, the GeoNetworking packet is secured by digital signatures and certificates, which are included in the header of the GeoNetworking secured packet. The extended header includes information for SHB, GBC, or TBC. The optional payload includes a V2X message payload such as CAM, DENM, MAPEM, SPATEM.

| | | | |
|---|---|---|---|
| **MAC Header** | **LLC Header** | **GeoNetworking Basic Header** | **GeoNetworking Secured Packet with GeoNetworking Common Header, Optional Extended Header and Optional Payload** |

Figure 19. *GeoNetworking packet structure (derived from [63]).*

### 5.1.2 Fields of the basic header

The fields of the basic header are shown in 14. Whether the security is enabled or not is identified by the value of NH (Next Header). The RHL (Remaining Hop Limit) field is significant in the geographical forwarding function described in Section 4.3. When a node receives a GeoNetworking packet, the value of RHL is decremented. The node can forward the packet only if the value of RHL is not 0. LT (Lifetime) field is also significant.

| **Field Name** | **Description** |
|---|---|
| Version | Version of GeoNetworking Protocol |
| NH (Next Header) | The type of header which follows Basic Header. If security is enabled, the value of this field is set to 2 (secure header). On the other hand, if security is not enabled, the value is set to 1 (common header). |
| Reserved | Reserved field for future use. Set to 0. |
| LT (Lifetime) | Maximum tolerable time a packet can be buffered. The packet which cannot reach destination within the time specified in this field shall not be processed. |
| RHL (Remaining Hop Limit) | The field used in geographical forwarding function of GeoNetworking protocol. Decremented by 1 for each time when a node receives the packet. If the RHL become 0, the packet shall not be forwarded. |

Table 14. *Fields of Basic Header.*

GeoNetworking packet shall be reached the destination within the value of the LT field. The traffic condition represented by the positions of vehicles and events happening on the road is changed so dynamically. Therefore, if a large value is assigned to the LT field, the received packet may be meaningless and misleading for the receiver.

### 5.1.3 Fields of the common header

The fields of the common header are shown in 15. The HT field indicates the forwarding scheme such as GBC, TSB and SHB. Thus, the receivers of a GeoNetworking packet can determine which forwarding scheme is applied for the packet. Along with HT, the HST field specifies a sub-type of the forwarding scheme. The shape of the destination area in GBC and whether multi-hop forwarding is allowed or not in TSB is indicated by this field. Besides, the MHL field works with HT and HST to determine the maximum hop limit. The MHL is always 0 when SHB (Single-Hop Broadcast). For other cases, the value of MHL is varied and determines how many hops are allowed in the GeoNetworking packet forwarding.

### 5.1.4 Fields of extended header

The fields of the extended header depend on the type of the header, such as TSB and GBC. The structure of the extended header of the TSB packet and GBC packet are specifically explained.

The extended header of TSB packet is shown in Table 16. The SN field is used for duplicated packet detection (DPD). An ITS-S can receive multiple copies of the same

| **Field Name** | **Description** |
|---|---|
| NH (Next Header) | The type of header which follows Basic Header. Typically, it is a type of transport layer. |
| Reserved | Reserved field for future use. Set to 0. |
| HT (Header Type) | The type of forwarding scheme such as GBC (GeoBroadCast) and TSB (Topologically-Scoped Broadcast). The value is set to 4 for GBC, while the value is set to 5 for TSB. |
| HST (Header Sub Type) | The sub-type of the forwarding scheme specified in the HT field. The sub-types of TSB have GEOBROADCAST_CIRCLE, GEOBROADCAST_RECT and GEOBROADCAST_ELIP, each of which means the shepe of the destination area is circle, rectangle and ellipse, respectively. The sub-types of TSB are SINGLE_HOP and MULTI_HOP. The MHL (Maximum Hop Limit) field in the common header is always 0 when the sub-type is SINGLE_HOP. |
| TC (Traffic Class) | This field is related to link layer (Layer 2). For instance, whether the packet shall be buffered or not if the neighbor does not exists can be specified here. |
| Flags | The type of the ITS-S. If the ITS-S is mobile (*e.g*. a vehicle), the value of this field is set to 0. On the other hand, the node does not move (*e.g*. a RSU), the value of this field is set to 1. |
| PL (Payload Length) | Length of GeoNetworking payload. Typically, the sum of data length of transport layer and application layer. |
| MHL (Maximum Hop Limit) | The maximum hop limit applied to the geographical forwarding function. The value is constant. Therefore, the value of MHL is not decremented by ITS-Ss that forward the GeoNetworking packet. |
| Reserved | For future usage. Set to 0. |

Table 15. *Fields of Common Header.*

| Field Name | Description |
|---|---|
| SN (Serial Number) | The index of the transmitted TSB packet, which is used to detect duplicated GeoNetworking packets. |
| Reserved | Reserved field for future use. Set to 0. |
| SO PV (Source Position Vector) | The position, state and timestamp of the source. |

Table 16. *Fields of Extended Header in TSB packet.*

| Field Name | Description |
|---|---|
| GN_ADDR (GeoNetworking Address) | The network address for the node . |
| TST(Timestamp) | The time when the latitude and longitude of the node is acquired. The time is the elapsed TAI (Temps Atomique International) as known as International Atomic Time since 2004-01-01 00:00:00.000 UTC. |
| Lat (Latitude) | Latitude of the node position expressed in 1/10 micro degree. |
| Lon (Longitude) | Longitude of the node position expressed in 1/10 micro degree. |
| PAI (Position Accuracy Indicator) | Position accuracy indicator of the node position. |
| Speed | Speed of the node with the units of 0.01 meter per second. |
| Heading | Heading of the node with the units of 0.1 degree from the north. |

Table 17. *Fields of Long Position Vector.*

packet due to the forwarding of the packet. If an ITS-S handles the multiple copies as the different ones, the ITS-S would misunderstand the current traffic state. In this regard, the DPD is significant. Algorithms for DPD include (1) sequence number and timestamp-based one and (2) timestamp-based one [63]. In Vanetza [55], one of the implementations of the ETSI protocol stack, sequence number and timestamp-based algorithm is adopted. The SO PV field is significant. It determines the source position, the timestamp when the source position is acquired and the current state of the source, such as speed and heading. Besides, the network address allocated for the node is also contained. The complete list of fields included in PV is shown in 17. The PV includes long position vector used for the position vector of a source and short position vector typically used for the position vector of a destination. SO PV of TSB and GBC do not need the position of a destination. Therefore, only the structure of the long position vector is shown in this thesis.

Some of the fields of the extended header in the GBC packet are the same as the fields in the TSB packet. But, several fields for determining the shape of the destination area are appended. The list of fields of the extended header in the GBC packet is shown in Table 18. With the value of fields GeoAreaPosLatitude, GeoAreaPosLongitude, Distance $a$, Distance $b$ and Angle $\theta$, the shape of the destination area is identified as shown in Figure 20 (derived from [63]). The shape can be a circle, rectangle or ellipse, determined by the HST field of the common header. Point A is the point determined by GeoAreaPosLatitude and

| Field Name | Description |
|---|---|
| SN (Serial Number) | The index of the transmitted GBC packet, which is used to detect duplicated GeoNetworking packets. |
| Reserved | Reserved field for future use. Set to 0. |
| SO PV (Source Position Vector) | The position, state and timestamp of the source. |
| GeoAreaPosLatitude | The latitude of center position of the destination area. |
| GeoAreaPosLongitude | The longitude of center position of the destination area. |
| Distance $a$ | Distance $a$ of the geometric shape. |
| Distance $b$ | Distance $b$ of the geometric shape. |
| Angle $\theta$ | Angle of the geometric shape. |
| Reserved | Reserved field for future use. Set to 0. |

Table 18. *Fields of Extended Header in GBC packet.*

| BTP Port | ITS facilities layer entity |
|---|---|
| 2001 | CA |
| 2002 | DEN |
| 2003 | RLT |
| 2004 | TLM |

Table 19. *The port numbers for CAM, DENM, MAPEM and SPATEM.*

GeoAreaPosLongitude.

## 5.2 BTP

BTP (Basic Transport Protocol) is the transport layer protocol in the ETSI protocol stack, which enables message multiplexing/demultiplexing between the ITS facilities layer (application layer) and GeoNetworking layer. The BTP protocol multiplexes/demultiplexes the V2X messages from/to different processes at the ITS facilities layer. A fixed port is assigned to an ITS facilities layer entity such as CA and DEN so that the ITS-S can enable or disable services by opening/closing ports. The port numbers for CA, DEN, RLT and TLM, which are the services this thesis focuses on as shown in Table 4, are available at Table 19.

BTP has two types of protocol headers, namely BTP-A and BTP-B. The BTP-A is for interactive packet transport, whereas the BTP-B is for non-interactive packet transport. Interactive transport means that the source expects a response from the destination. Therefore, BTP-A includes a source port, while BTP-B does not include a source port. In most cases of V2X communication, a response from the destination is not expected like UDP. Therefore, the BTP-B type is usually used. The structure of the BTP-B header is shown in Table 20. Both the BTP-A header and BTP-B header consist of 32 bits. In the BTP-A header, the destination port info field is removed, and the source port is added instead.

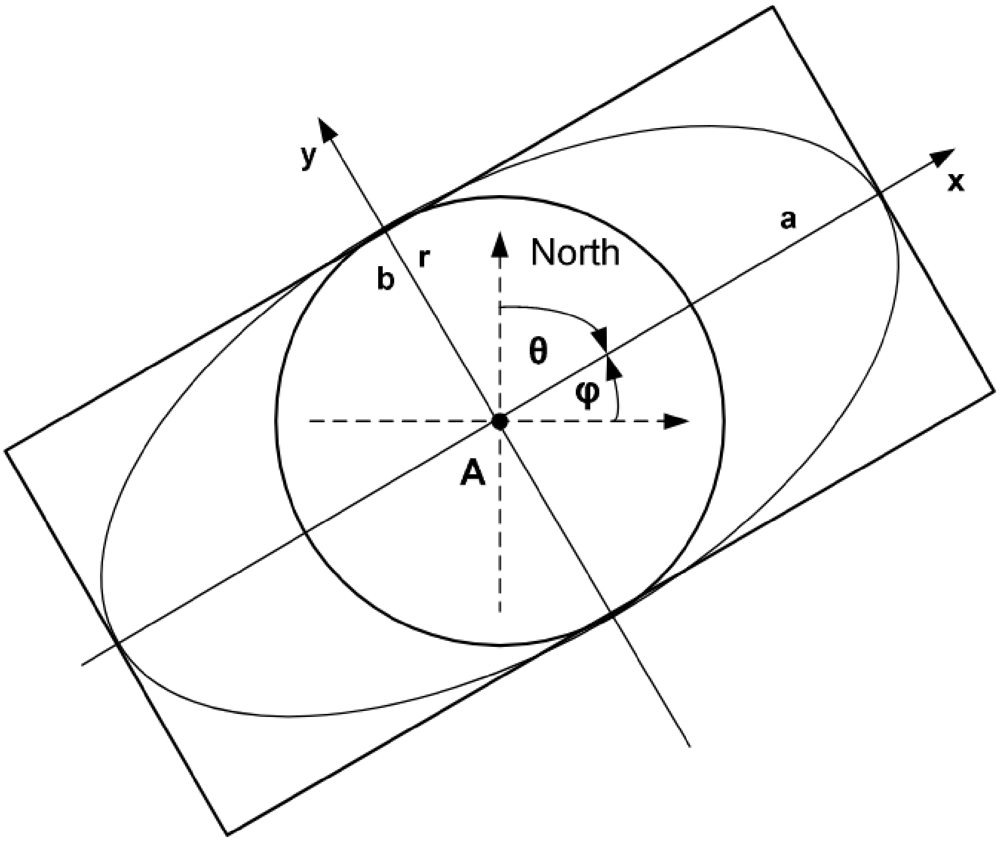


Figure 20. *The usage of parameters to determine the shape of destination area (derived from [63]).*

| **Field Name** | **Description** |
|---|---|
| Destination port | Represents the protocol entity at the ITS facilities layer such as CAM and DENM. |
| Destination port info | Additional information about destination port. Typically, this field is set to 0, which represents no additional information. |

Table 20. *Fields of BTP-B header.*

## 5.3 CAM

### 5.3.1 Overview

In this section, the structure of CAM (Cooperative Awareness Message) is specifically explained among CAM, DENM, MAPEM and SPATEM based on [44]. CA (Cooperative Awareness) service is an ITS facilities layer entity that enables cooperative awareness among road users to inform each other's position and dynamics [44] The information is packed up to CAM (Cooperative Awareness message). The content of CAM depends on the type of road users. For cars, the content includes a motion state that dynamically changes with high frequency and the basic information of the vehicle, such as the type of vehicle. For special vehicles such as emergency cars and vehicles for public transport, additional information can be added depending on the type of the special vehicles.

### 5.3.2 Overall structure of CAM

The structure of CAM is shown in Figure 21 (derived from [44]). ITS PDU (Protocol Data Unit) header is the common header for all ITS facilities layer messages, which includes protocol version, message ID and station ID. The basic container and HF (High Frequency) container are mandatory containers to be included, whereas LF (Low Frequency) container and special vehicle container are optional that can be added depending on the type of vehicle.

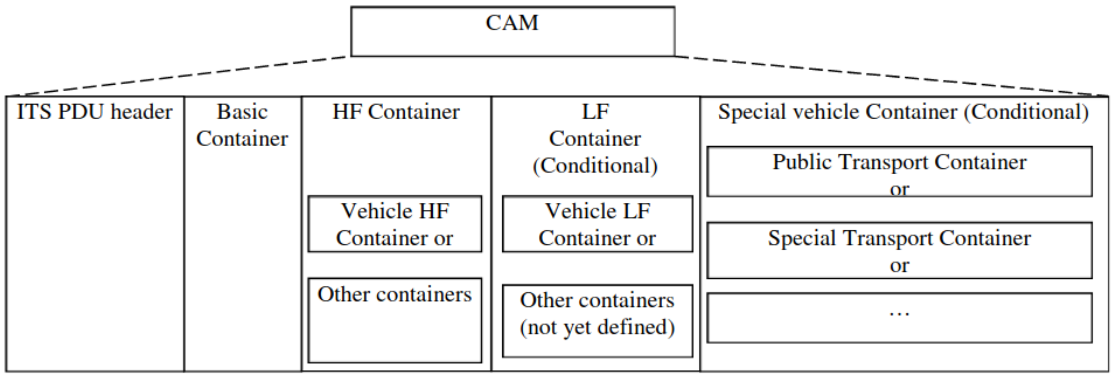

CAM
ITS PDU header
Basic Container
HF Container
Vehicle HF Container or
Other containers
LF Container (Conditional)
Vehicle LF Container or
Other containers (not yet defined)
Special vehicle Container (Conditional)
Public Transport Container or
Special Transport Container or
...

Figure 21. *General Structure of CAM (derived from [44]).*

The objective of this thesis is to develop IDS/IPS for the ETSI ITS-G5 V2X communication. Therefore, the important fields to determine whether messages can be processed or not are mandatory ones, which include basic information about the messages. For that reason, only the mandatory fields are explained.

| Field Name | Description |
|---|---|
| Protocol Version | Version of the ITS Message. |
| Message ID | The type of the ITS message represented by an 8 bit unsigned integer including denm(1), cam(2), spatem(4) and mapem(5). |
| Station ID | The identifier of the ITS-S that generates the ITS message represented by a 32 bit unsigned integer. |

Table 21. *Fields of ITS PDU Header.*

| Field Name | Description |
|---|---|
| Station Type | The type of the station represented by an 8 bit unsigned integer including pedestrian(1), cyclist(2), moped(3), motorcycle(4), passengerCar(5), bus(6), lightTruck(7), heavyTruck(8), trailer(9), specialVehicles(10), tram(11) and roadSideUnit(15). |
| Reference Position | the position of the station represented by latitude, longitude and altitude with their confidence. |

Table 22. *Fields of Basic Container in CAM.*

### 5.3.3 Fields of ITS PDU header

The fields of the ITS PDU header are the same among all ITS facilities layer messages, namely CAM, DENM,MAPEM and SPATEM. It is included at the beginning of an ITS message as the message header. The fields of the ITS PDU header are available at Table 21.

### 5.3.4 Fields of basic container

The fields of basic container are shown in Table 22. The basic container has basic information about the ITS station, namely the type of ITS station and its position.

### 5.3.5 Fields of HF Container

The fields of HF (High Frequency) container are shown in Table 23. The HF container contains motion state, which should be updated with high frequency. The generation rate $T_{CamGen}$ shall be $1\ Hz \leq T_{CamGen} \leq 10\ Hz$ [44]. With the generation rate $T_{CamGen}$, the information included in the HF container is updated.

## 5.4 Example of GeoNetworking packet

In this section, an example of a secured GeoNetworking packet consisting of GeoNetworking header, BTP header and ITS message in case the ITS message is CAM is shown. The

| **Field Name** | **Description** |
|---|---|
| Heading | Heading of the vehicle with its confidence. |
| Speed | Speed of the vehicle with its confidence. |
| Drive Direction | The direction of the vehicle's motion, forward, backward or unavailable. |
| Vehicle Length | The length of the vehicle. |
| Vehicle Width | The width of the vehicle. |
| Curvature | The curvature of the vehicle with its confidence. |
| Yaw Rate | The yaw rate of the vehicle with its confidence. |

Table 23. *Fields of HF Container in CAM.*

structure of GeoNetworking header, BTP header and CAM is described in Section 5.1, Section 5.2 and Section 5.3, respectively. Listing 5.1 shows the example in YAML format.

```
---
GeoNetworking_CW: Secured (TSB Single Hop)
  Basic Header
    0001 .... = Version: 1
    .... 0010 = Next Header: Secured (2)
    Reserved: 0
    Lifetime 60000 ms
      0001 10.. = Multiplier: 6
      .... ..10 = Base: 10 s (2)
    Remaining Hop Limit: 1
  Secure Header
    Version: 2
    Header Length: 184
    Header Fields
      Header Field: Signer Info (128)
      SignerInfo Type: certificate (2)
      Certificate
        Version: 2
        SignerInfo Type: certificate digest with sha256 (1)
        Hashed8: 0xfcb6d18273246b5e
        Subject Info: authorization ticket (1)
        IntX: 0
        Subject Attribute
          Subject Attr Len: 82
          Subject Attribute Type: verification key (0)
          Public Key
          Subject Attribute Type: assurance level (2)
          Assurance Level: 0x00
          Subject Attribute Type: its aid ssp list (33)
          IntX: 11
          ITS AID: CAM (0x00000024)
          IntX: 3
          SSP: 010000
          ITS AID: DENM (0x00000025)
          IntX: 4
          SSP: 01000000
        IntX: 9
        Validity Restriction: time start and end (1)
        Start Time: 2022-02-06 11:16:42 (571231007)
        End Time: 2022-02-13 12:16:42 (571839407)
        Signature
          Public Key Alg: ecdsa nistp256 with sha256 (0)
          ECC Point Type: x-coordinate only (0)
          Opaque: b242b6e0500dbda3a9e347d774038076586eaa6bd78a8afb
          Opaque: 05a9019e85a7ed5e7e2c4160ff30f297ebabdc124e2897b6
      Header Field: Generation Time (0)
      Generation Time: 2022-02-06 14:23:29.121503 (571242214121503)
      Header Field: ITS AID (5)
      ITS AID: CAM (0x00000024)
    Payload Type: signed (1)
    Payload Data Length: 81
  Common Header
    0010 .... = Next Header: BTP-B (2)
    .... 0000 = Reserved: 0
    0101 .... = Header Type: TSB (5)
    .... 0000 = Header Subtype: Single Hop (0)
    Traffic Class: 0x00
      0... .... = Store-Carry-Forward: 0
      .0.. .... = Channel Offload: 0
      ..00 0000 = DP ID (DCC Profile Id): 0
    Flags: 0x80
      1... .... = Mobile Flag: Mobile (1)
      .000 0000 = Reserved: 0x00
    Payload Length: 45
    Maximum Hop Limit: 1
    Reserved: 0
  Topology-Scoped Broadcast
    Source Position Vector
      GN Address: 0x80000a0027000000
      Timestamp: 11563.705s (11563705)
      Latitude: 48.7668616 (48.7668616 deg) (487668616)
      Longitude: 11.4320679 (11.4320680 deg) (114320679)
      1... .... .... .... = PAI: 1
      .000 0000 0000 0000 = Speed: 0x0000 (0.00 m/s) (0.00 km/h) (0)
      Heading: 0.0 deg (0)
    Reserved (G5 DCC)
  Secure Trailer
    Trailer Length: 67
    Trailer Type: signature (1)
    Signature
      Public Key Alg: ecdsa nistp256 with sha256 (0)
      ECC Point Type: x-coordinate only (0)
      Opaque: dbce3efdd05803b7bdf3b4cb8e449cb4215abff7b877bd16
      Opaque: 5236053fc6bafe61936d45f020303e370d72b3996d96ea18
Basic Transport Protocol (Type B)
  Destination Port: 2001
  Destination Port Info: 0
ETSI ITS (CAM)
  CAM
    header
      protocolVersion: 2
      messageID: cam (2)
      stationID: 1 (0x00000001)
    cam
      generationDeltaTime: Unknown (29417) (0x72e9, 29.417 sec)
      camParameters
        basicContainer
          stationType: passengerCar (5)
          referencePosition
            latitude: Unknown (487668620) (48.7668620 deg)
            longitude: Unknown (114320680) (11.4320680 deg)
            positionConfidenceEllipse
              semiMajorConfidence: unavailable (4095)
              semiMinorConfidence: unavailable (4095)
              semiMajorOrientation: unavailable (3601)
            altitude
              altitudeValue: unavailable (800001)
              altitudeConfidence: unavailable (15)
        highFrequencyContainer: basicVehicleContainerHighFrequency (0)
          basicVehicleContainerHighFrequency
            heading
              headingValue: wgs84North (0) (0.0 deg)
              headingConfidence: equalOrWithinOneDegree (10)
            speed
              speedValue: standstill (0) (0.00 m/s, 0.00 km/h)
              speedConfidence: equalOrWithinOneCentimeterPerSec (1) (0.01 m/s)
            driveDirection: forward (0)
            vehicleLength
              vehicleLengthValue: unavailable (1023)
              vehicleLengthConfidenceIndication: noTrailerPresent (0)
            vehicleWidth: unavailable (62)
            longitudinalAcceleration
              longitudinalAccelerationValue: unavailable (161)
              longitudinalAccelerationConfidence: Unknown (0)
            curvature
              curvatureValue: straight (0)
              curvatureConfidence: unavailable (7)
            curvatureCalculationMode: yawRateUsed (0)
            yawRate
              yawRateValue: unavailable (32767)
              yawRateConfidence: degSec-000-01 (0)
```

Listing 5.1. Example of GeoNetworking Packet

# 6. System architecture

## 6.1 System overview and deployment

Figure 22 shows the system architecture of the proposed IDS/IPS. The IDS/IPS is network-based IDS/IPS. The IDS/IPS is deployed at OBU and connected with two network interfaces of the OBU. The source network interface is the one that captures packets on the IEEE 802.11p channel, whereas the destination interface is the one connected with ITS applications such as CA and DEN. Through the source interface, packets are acquired and then inspected by the IDS/IPS. Through the destination interface, the accepted packets are forwarded to ITS applications. The ITS applications transform data included in V2X messages and publish to ROS topics shared with the autonomous vehicle core process. A simple application has been developed by the author and is available at `https://github.com/camsenec/its_apps`.

Next, the internal filtering process in the IDS/IPS is explained:

1. The IDS/IPS acquires a packet from the buffer linked to a network interface.
2. The acquired packet is decoded and values for the fields as shown in the previous chapter are extracted. Also, the packet is classified based on the type of message included in the packet to pass it to a proper application layer parser. The application layer parser parses data on the application layer.
3. The detection module reads patterns in the packet and compares them to the detection rules defined in advance. Detection results are logged to log files.

Figure 23 shows the detailed architecture of the decode module. First, the decoder gets an ethernet frame captured at a network interface and stored on an input buffer. From the Ethernet frame, the ethernet header is extracted, and the GeoNetworking packet is passed to the next step. Next, from the GeoNetworking packet, the basic header, secure header, common header and extended header are decoded. In this section, significant fields for IDS/IPS are only explained. The full list of the GeoNetworking header fields is shown in Section 5.1.

The basic header includes LT (Lifetime) and RHL (Remaining Hop Limit). The se-

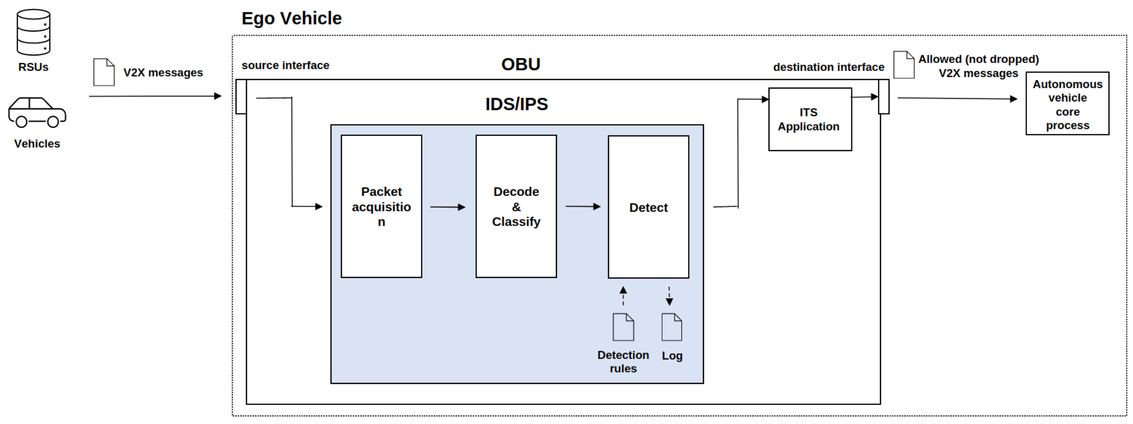


Figure 22. *System Overview and deployment of the IDS/IPS.*

cured header includes certificate and the type of certificate (`SignerInfoType`) such as `Certificate`, `Certificate_Digest_With_SHA256` and `Certificate_Chain` shown in Figure 7. The secure header is optional. If the V2X message has not been signed by an authorization ticket, the secured header is not included. In that case, the NH field of the basic header is set to the common header, while the field is set to the secure header if the secured header is included. Therefore, the NH field of basic header can be used to detect unauthorized messages. The common header is the most significant header to detect malicious messages. HT (Header Type) represents the type of the forwarding scheme, such as GBC and TSB. Others include HST (Header Sub Type), MHL (Maximum Hop Limit) and Flags. HT and MHL can be inspected to prevent attacks using AV 5. The extended header is also important, especially when the forwarding scheme is GBC because the destination area in the GBC packet is included in the extended header. Regardless of the forwarding scheme, the extended header includes SO PV, which indicates the position and GeoNetworking address of the sender.

After the GeoNetworking packet is decoded, the BTP datagram is extracted. The destination port can be the hint for classifying the V2X message. The classification process is explained later in this section. Finally, ITS PDU is obtained by removing the BTP header from the BTP datagram. The message ID indicates the type of the V2X message, such as CAM, DENM, SPATEM and MAPEM. Station ID can be treated as an identifier of the origin of the message in addition to the GeoNetworking address.

Figure 24 shows architecture of the classification module. The classification module classifies V2X messages based on the value of the destination port and passes it to the appropriate application layer parser. The application layer parser immediately drops the message if the type of V2X message indicated by the message ID does not correspond to the type of the application layer parser. The ports are not physical ports but virtual ports,

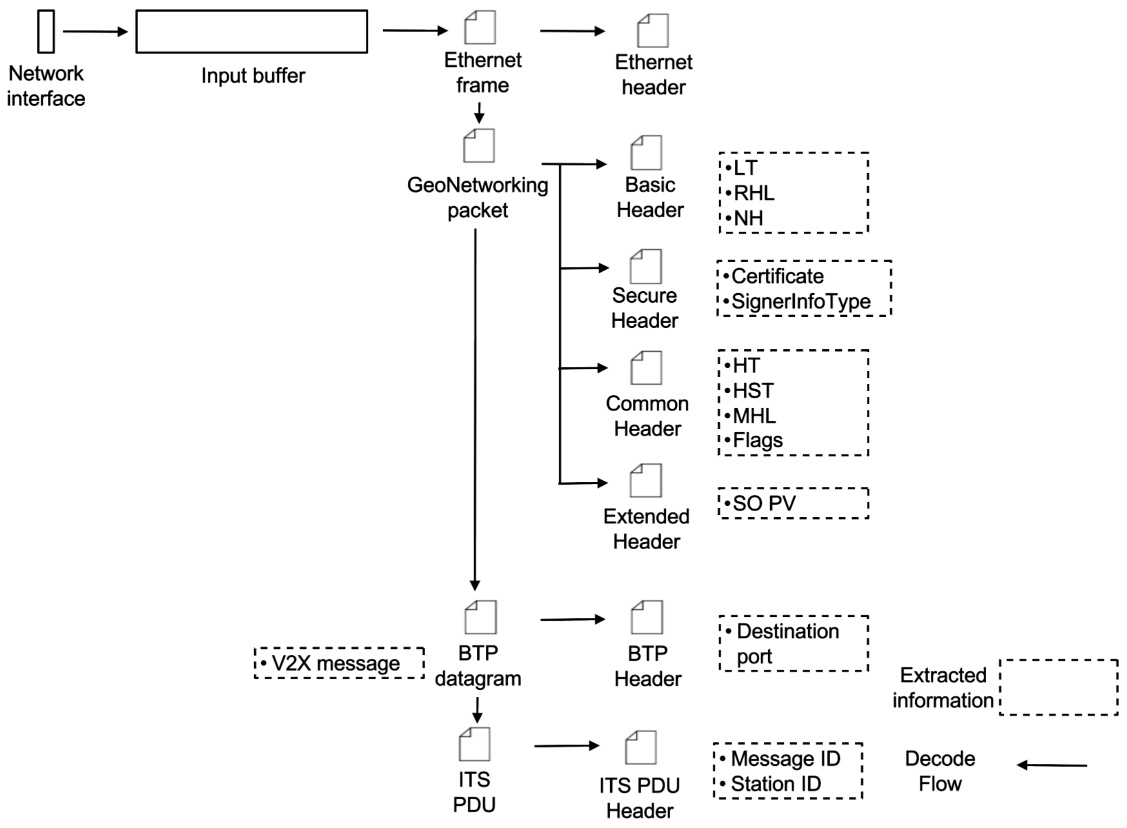


Figure 23. *Decoder.*

which means that the ports are not bound to sockets.

## 6.2 Functional requirements

Based on the discussion for countermeasures in Section 4.8, the minimal Functional Requirements (FRs) of the IDS/IPS can be listed as follows:

- *FR-1* The IDS/IPS can block unauthorized messages.
- *FR-2* The IDS/IPS can block TSB/GBC packets with malicious configuration.
- *FR-3* The IDS IPS can only allows messages coming from trusted ITS-Ss.

For unauthorized message blocking, the NH field of the basic header can be used. If the value of the NH field is common header, the message is not authorized.

Malicious configuration of TSB/GBC packet includes the case when maximum hop limit is large and the case when the position of the source which generated the packet is far from the ego-vehicle. For the first case, if the value of the HT field is GBC or TSB and the value of the MHL field is larger than some threshold, the message should be blocked. For the second case, if a vehicle does not need information from distant ITS-Ss, the vehicle can

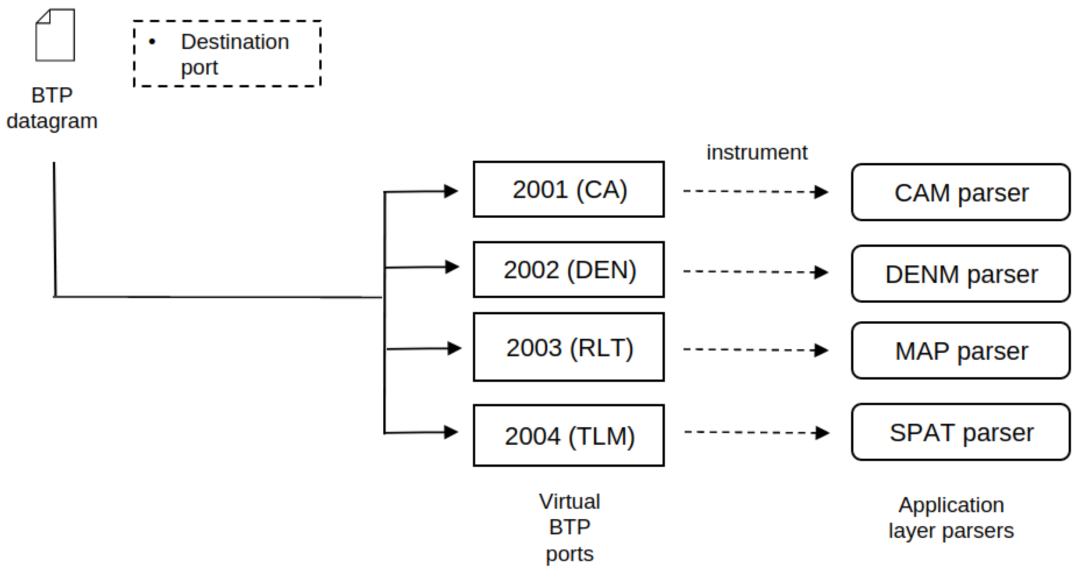


Figure 24. *Classifier.*

block messages coming from ITS-Ss outside of a limited area. The area, for instance, can be defined as a circle with a radius of $r$ m centered on the ego-vehicle. The position of a sender is available as the SO PV (Source Position Vector) in the extended header of the GeoNetworking packet. The problem with this message blocking outside of an area is that there is a possibility that a malicious vehicle/person exists in the vicinity. However, the period of time to receive malicious messages is not long because the area to restrict incoming messages moves as the ego-vehicle moves. Let's consider the case shown in Figure 25. In Figure 25, the area for message restriction is set to the circle with a radius of $100$ m centered on the ego-vehicle. Assuming that a malicious person is in the area for message restriction, the maximum period to receive messages from the malicious person can be calculated as follows:

$$100 \times 2 \times \frac{3600}{36 \times 1000} = 20\ s \tag{6.1}$$

, where the diameter of the circle is calculated by $100 \times 2$ and the speed of the vehicle is calculated by $\frac{36 \times 1000}{3600}$ m/s. If the radius of the circle is reduced to $10$ m, then the period to receive messages from the malicious person is also reduced to $2$ s. The risks that are attributed to delivering malicious messages to an arbitrary area, such as map database poisoning and notification of false information, cannot be mitigated completely. However, blocking messages coming from far from the ego-vehicle is still an effective method.

Only allowing messages coming from trusted ITS-Ss is the countermeasure to prevent the false signature flood. Several ways can be adopted to allow messages from trusted

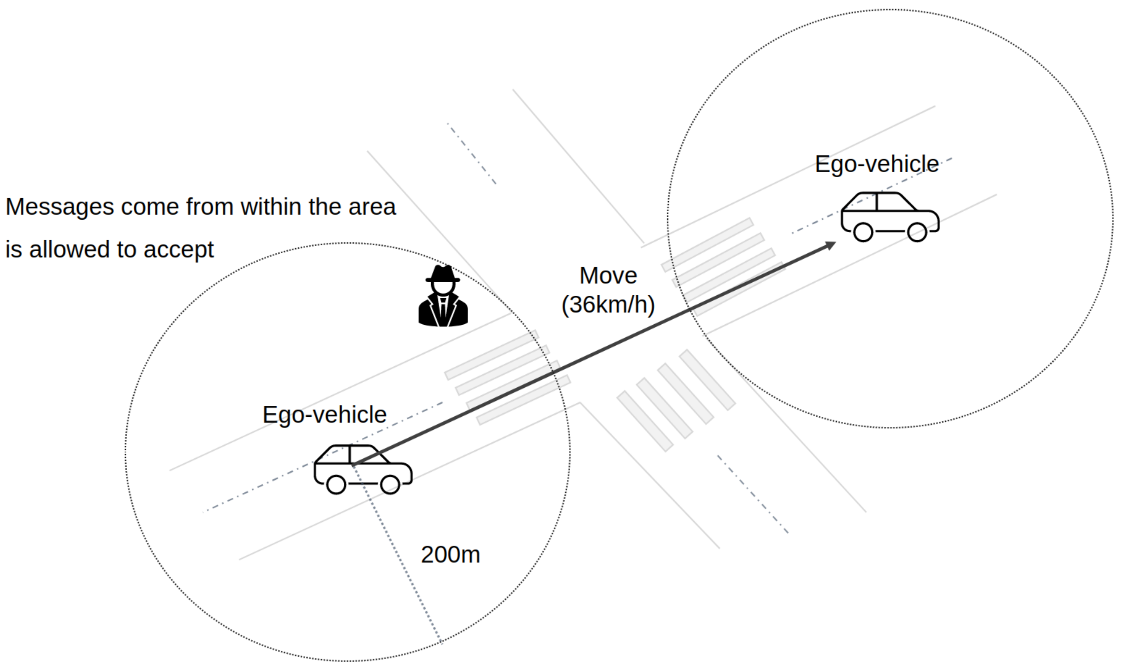


Figure 25. *A moving vehicle and the area to restrict incoming messages.*

ITS-Ss depending on what kinds of ITS-Ss are considered “trusted”. The simplest way to define trusted ITS-Ss is to use identities such as GN_ADDR (GeoNetworking address) and Station ID. By listing trusted ITS-Ss by GeoNetworking addresses or Station IDs in an IDS/IPS rule, the IDS/IPS can determine whether the message comes from a trusted ITS-S by comparing Station ID and GeoNetworking address of an incoming message with the listed GeoNetworking addresses and Station IDs. The problem with this way is GeoNetworking address and Station ID are changed periodically. If the GeoNetworking address or Station ID of a trusted ITS-S is updated, the GeoNetworking address or Station ID stored in IDS/IPS rule should also be updated. Therefore, the distribution center, which is one of the countermeasures discussed in Section 4.8 should be operated to notify updates of identities from EA to ITS-Ss. To define ITS-Ss in the vicinity as trusted is another way, which can be achieved by the same method as the one to block messages generated far from the ego-vehicle. Only allowing messages that come from ITS-Ss in the vicinity can decrease the probability of DoS. However, for the same reason why risks attributed to delivering malicious messages to arbitrary areas cannot be mitigated completely, the probability of false signature flood-based DoS cannot be mitigated completely.

In summary, the methods to satisfy the functional requirements can be described as Table 24. The implementation of the proposed IDS/IPS makes it possible to enable these methods. This means that the proposed IDS/IPS should allow developers of the V2X application – utilization of V2X messages for core processes of autonomous vehicles, to enable/disable the features achieved by the methods following their intension.

| **Functional requirement** | **Looked up fields** | **Method description** |
|---|---|---|
| FR 1 | NH | Messages whose NH header is common header are blocked. |
| FR 2 (maximum hop limit) | HT, MHL, destination area | Messages whose HT header is GBC and MHL or destination area is larger than some threshold are blocked. |
| FR 2 (destination area) | SO PV | Messages coming from ITS-Ss within some area are allowed, where the position of ITS-Ss is determined by SO PV. |
| FR 3 (Identity based) | GN_ADDR, station ID | Messages with GN_ADDR or station ID which corresponds to one of the identity listed in IDS/IPS rule are allowed. |
| FR 3 (Area based) | SO PV | Messages coming from ITS-Ss within some area are allowed, where the position of ITS-Ss is determined by SO PV. |

Table 24. *Methods to satisfy functoinal requirements.*

# 7. Implementation

In this chapter, the implementation of the proposed IDS/IPS specifically to satisfy functional requirements shown in Table 17.

## 7.1 Overview

The proposed IDS/IPS is named Mitvane. Mitvane is IDS (Intrusion Detection System) / IPS (Intrusion Prevention System) for ETSI-G5-based V2X communication. The Mitvane was developed in object-oriented design so that additional detection modules can be integrated easily. The Mitvane is available as OSS `https://github.com/camsenec/Mitvane`. The Mitvane is licensed under LGPLv3 [65]. The proposed IDS/IPS is written in C++ and built on the CMake build system. The IDS/IPS depends on several libraries such as Vanetza [55], Boost [66] and yaml-cpp [67]. Vanetza is one of the implementations of ETSI ITS-G5 protocol stack and offers Vanetza API. The Vanetza API can be used to achieve various tasks such as decoding and verification of messages. Boost has several components to enable advanced functions such as asynchronous packet acquisition (Boost::Asio), logging (Boost::log) and parsing of command-line arguments (Boost::program_options). The yaml-cpp is a YAML parser and emitter in C++ matching YAML 1.2 spec [68].

## 7.2 Detection rule

First of all, a description of detection rules is needed. Listing 7.1 shows an example of a file describing detection rules. The detection rule file follows YAML format.

```yaml
---
# action: "drop", "alert"
# protocol: "GeoNetworking", "BTP", "Facility", "CA", "DEN", "RLT", "TLM"

rules:
  # Rule for FR1
  - action: "drop"
    protocol: "GeoNetworking"
    meta:
      msg: "Unauthorized message is detected."
      sid: 1
      rev: 1
    nh: "common"

```

```yaml
# Rule for FR2 (maximum hop limit)
- action: "drop"
  protocol: "GeoNetworking"
  meta:
    msg: "GBC with large max hop limit is detected."
    sid: 2
    rev: 1
  ht: "GBC"
  mhl: 3

# Rule for FR2 (destination area)
- action: "drop"
  protocol: "GeoNetworking"
  meta:
    msg: "GBC from a source far from the ego-vehicle is detected"
    sid: 3
    rev: 1
  ht: "GBC"
  allowed_so_range:
    shape: "circle"
    distance_a: 100
    distance_b: 100

# Rule for FR3 (Identity based)
- action: "drop"
  protocol: "facility"
  meta:
    msg: "A packet from not trusted ITS-S is detected. (Identity based)"
    sid: 4
    rev: 1
  allowed_ids:
    - 123490
    - 123491
    - 123492

# Rule for FR3 (Area based)
- action: "drop"
  protocol: "GeoNetworking"
  meta:
    msg: "A packet from not trusted ITS-S is detected. (Area based)"
    sid: 5
    rev: 1
  allowed_so_range:
    shape: "circle"
    distance_a: 100
    distance_b: 100
```

Listing 7.1. Detection rules

Five rules are stated in the detection rule file. Each of the rule corresponds to a functional requirement shown in Table 14. All of the rules are required to have `action`, `protocol`, `meta` and detection patterns.

The `action` determines what happens when the rule matches. The current supported actions are

- `alert`,
- `drop`.

The `alert` generates an alert when a packet matches one of the patterns defined by rules. On the other hand, the `drop` drops the packet while generating an alert. The action is perfomed only when all the patterns in a signature matches patterns in a received packet. The `protocol` tells the IDS/IPS which protocol it concerns. Currently, the supported protocol includes

- `GeoNetworking`,
- `BTP`,
- `CA`,
- `DEN`,
- `RLT`,
- `TLM`,
- `Facility`

`GeoNetworking` and `BTP` are the networking layer and the transport layer protocol of the ETSI ITS-G5 protocol stack, respectively, as shown in Figure 1. `CA`, `DEN`, `RLT` and `TLM` corresponds to the services, entities of facility layer, shown in Table 4. The messages used by the services are CAM, DENM, MAPEM and SPATEM, respectively. When one of these protocols is specified, an application layer parser has to decode application layer data after the message is classified by the classifier shown in Figure 24. Finally, the `Facility` is specified to apply the detection rule for all types of V2X messages including `CA`, `DEN`, `RLT` and `TLM`. For example, the `Facility` is specified when a user wants to inspect station IDs in the ITS PDU header, (Table 21), which is included in all types of V2X messages.

The `meta` consists of three keywords, namely `msg`, `sid` and `rev`. The `msg` gives a message generated by an alert or drop of a packet when a packet matches one of the patterns defined by rules. The `sid` gives every rule (signature) its own id. In Listing 7.1, IDs from 1 to 5 are assigned to the rules. The `rev` represents the version of a rule. If a rule is modified, the number of rev will be incremented by the writer of the rule. In Listing 7.1, all rules has value 1 for `rev`.

Finally, detection patterns follows the `action`, `protocol` and `meta`. Detection patterns are represented by a key-value set, where the key is one of the fields explained in Chapter 5 and the value is a pattern to be detected. For instance, FR 1 in Table 24 need to look up NH field and if NH field is common header, the packet is blocked. The corresponding rule to the FR 1 can be written as `nh: "common"` with action `drop`. With the rule, packets whose NH field is common are detected and then dropped.

The value for detection patterns can also be key-value set or sequence. For example, the rule for FR 2 (destination area) has `allowed_so_pv` keyword. The value for the `allowed_so_pv` is also written as a key-value set. The `distance_a` and `distance_b` is the parameter which determines the size of the limited area as shown in Figure 20. In the detection rule in Listing 7.1, packets from ITS-Ss outside of a circle with radius 100 m is blocked. On the other hand, the detection pattern `allowed_ids` for FR3 (Identity based) is stated as a sequence. The `allowed_ids` has a list of IDs of ITS-Ss whose generated packets are allowed to accept.

The keywords used in the example detection rule for file for describing detection patterns are summarized in Table 25.

For implementation, a C++ library yaml-cpp [67] is used to load YAML file and parse the file. A rule is stored to `Signature` object shown in Listing 7.2. The `signature` objects are managed by a map called `signatures`, where the key is one of the protocol supported by the proposed IDS/IPS and the value is a list of signatures for the protocol.

```cpp
enum class Action
{
    Drop, Alert, NotSupported_Action
};

enum class Protocol
{
    GeoNetworking, BTP, Facility, CA, DEN, MAP, SPAT, NotSupported_Protocol
};


struct MetaData
{
    std::string msg; uint32_t sid; uint16_t rev;

};

typedef boost::variant<
    int, std::string, geonet_destination_area, geonet_allowed_so_range,
        facility_ids_type
    > pattern_type;


struct Signature
{
    Action action; Protocol protocol; MetaData meta;
    std::map<std::string, pattern_type> patterns;

};
```

| **Protocol** | **Keyword** | **Description** |
|---|---|---|
| GeoNetworking | `nh` | The `nh` keyword is used to check whether a GeoNetworking packet is secured (signed) or not. The value can be `"common"` or `"secure"`. Typically, `"common"` is specified to detect or block not secured messages. |
| | `ht` | The `ht` keyword is used to check for a specific type of forwarding schemes (Header Type) of GeoNetworking. The supported types of forwarding schemes are `"GBC"`, `"TSB"` and `"SHB"`. For example, in the rule for FR2, `"GBC"` is specified to detect GBC packets. |
| | `mhl` | The `mhl` keyword is used to check the maximum hop limit in TSB and GBC. If the MHL of a received packet is larger than the value specified to `mhl` in the detection rule file, the corresponding action is performed under the condition that the other patterns also matches. |
| | `allowed_so_range` | The `allowed_so_range` keyword is used to check the source position vector. The keywords which consists of `allowed_so_range` are `shape`, `distance_a` and `distance_b`. With the values of these field, a range can be defined. If the source position vector does not exists in the defined range, the corresponding action is performed under the condition that the other patterns also matches. |
| Facility, CA, DEN, TLM, RLT | `allowed_ids` | The `allowed_ids` includes IDs of the ITS-Ss whose generated packets are allowed to accept. If a message with a station ID which is not included in the list of `allowed_ids`, the corresponding action is performed under the condition that the other patterns also matches. |

Table 25. *Keywords used in the example detection file.*

Listing 7.2. `Signature` object

## 7.3 Detailed design

The object which is in charge of packet acquisition, decoding, classification and detection is `idps_context`. The `idps_context` object receives signatures (rules) from rule_reader, extract patterns from packets and detects malicious packet based on the signatures. The `idps_context` instantiates objects such as decoder, classifier, application layer parser and detector and calls methods associated to the objects interacting with other components. Each object has a method to perform a specific task such as decoding and classification and return objects to `idps_context` needed for the following steps. The interactions between `idps_context` and other components, devices and main function are shown in Figure 26.

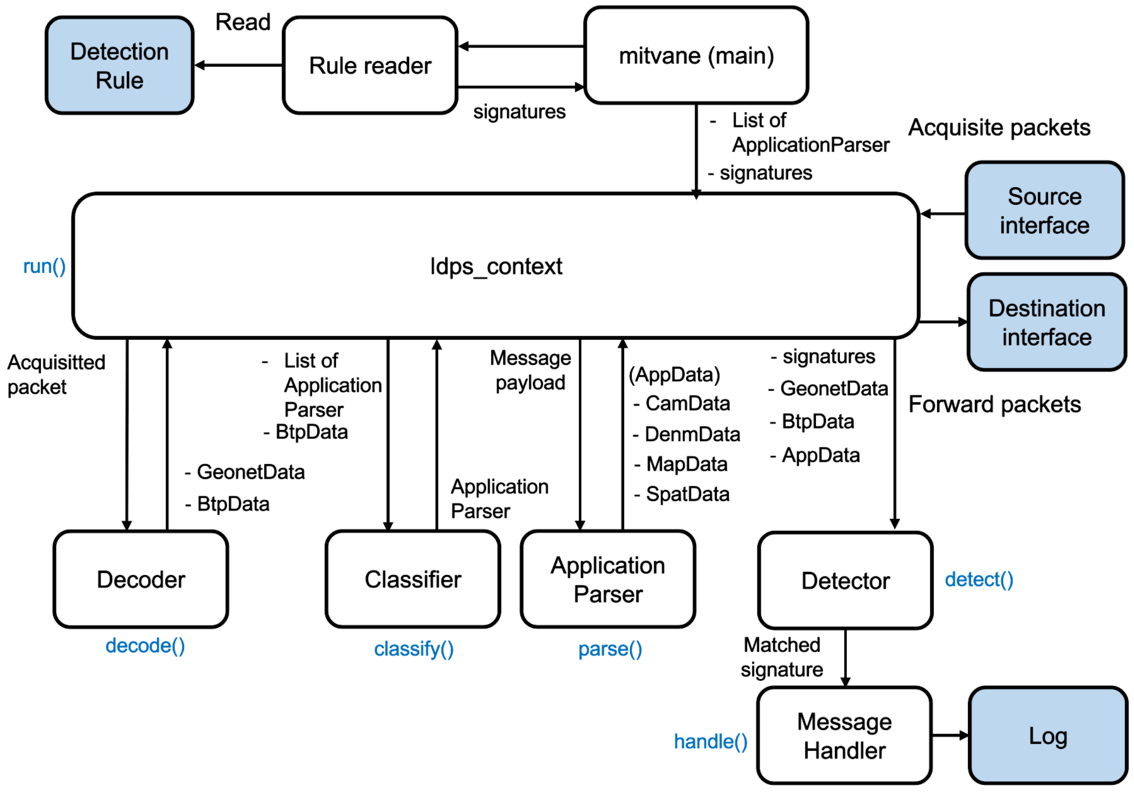


Figure 26. *Interactions of `idps_context` with other components.*

First, Mitvane instantiate `RuleReader` object and call `read()` function to read detection rule file. Then, the rules are returned to the main funcion as a map `signatures`, where the key is one of the protocols supported by the proposed IDS/IPS and the value is a list of rules for the protocol. Besides, the main function creates instances of application layer parsers (list of `ApplicationParser`). The parsers are passed to `idps_context` with the map `signatures`.

Next, every time source interface aquisites packet, `run()` call of `idps_context` is triggered. After `run()` call is triggered, `idps_context` creates `Decoder` instance and call decode() function, where the aquisited packet is passed as an argument. Then, the `Decoder` returns extracted data from the packet, namely `GeonetData` and `BtpData` to `idps_context`. The structure of `GeonetData` and `BtpData` are shown in Listing 7.3 and Listing 7.4. In short, `GeonetData` and `BtpData` holds patterns observed in a received packet including values required to conduct pattern matching for the keywords shown in Table 25.

```

struct GeonetData
{
    unsigned int protocol_version;
    vanetza::geonet::NextHeaderBasic next_header;
    vanetza::geonet::HeaderType header_type;
    boost::optional<vanetza::geonet::DestinationVariant> destination;
    vanetza::geonet::ShortPositionVector source_position;
    boost::optional<vanetza::security::HashedId8> certificate_id;
    boost::optional<vanetza::security::PayloadType> payload_type;
    boost::optional<uint8_t> secure_protocol_version;
    boost::optional<vanetza::security::SignerInfoType> signer_info_type;
    boost::optional<unsigned int> chain_size;
    vanetza::geonet::Lifetime remaining_packet_lifetime;
    uint8_t maximum_hop_limit;
    uint8_t remaining_hop_limit;
};
```

Listing 7.3. Structure of `GeonetData`

```
struct BtpData
{
    vanetza::geonet::UpperProtocol btp_type;
    boost::optional<vanetza::btp::port_type> source_port;
    vanetza::btp::port_type destination_port;
    boost::optional<decltype(vanetza::btp::HeaderB::destination_port_info)>
        destination_port_info;
};
```

Listing 7.4. Structure of `BtpData`

After the decoding, `idps_context` passes `BtpData` to `Classifier` and call classify() function, where the list of `ApplicationParser` is given as an argument. The `Classifier` classifies the received messages based on the destinaiton port number included in `BtpData` as shown in Figure 24 and returns the `ApplicationParser` corresponding to the type of message.

Next, `idps_context` calls parse() funciton of the parser returned by `Classifier`.

The passed argument is message payload, which is extracted in the decode process. Then, `AppData`, which is one of `CamData`, `DenmData`, `MapData` or `SpatData` is returned depending on the type of the message. The returned data always contain values of ITS IDS header (Table 21). Besides, the returned data includes additinal information depending on the type of message.

Finally, `idps_context` calls detect() function through `Detector` instance passing `GeonetData`, `BtpData`, `AppData` and `signatures` as well as the current position of the ego-vehicle. If the observed patterns matches one of the signatures, `handle()` call of `MessageHandler` is triggered. The `MessageHandler` performs the action for the matched rule and write the message to a log file.

Listing 7.5 shows the code of the `run()` funciton of `idps_context`. The `run()` decode the aquisited packet, classify the message, parse applicaition data and detect mathing patterns with the detection rule in order. The `detection_context` is a object which holds `GeonetData`, `BtpData` and `AppData` as well as message payload. In other words, the `detection_context` works as a bucket which stores data exchanged with other components.

```
void IdpsContext::run(vanetza::CohesivePacket&& packet, const vanetza::EthernetHeader&
    hdr)
{
    if (hdr.source != mib_.itsGnLocalGnAddr.mid() && hdr.type == vanetza::access::
        ethertype::GeoNetworking) {
        std::cout << "received packet from " << hdr.source << " (" << packet.size() << "
            bytes)\n";
        mitvane::DetectionContext detection_context;

        // 1. Decode
        std::unique_ptr<vanetza::PacketVariant> up_dec { new vanetza::PacketVariant(
            packet) };
        mitvane::GeonetDecoder decoder(mib_, detection_context);
        decoder.decode(std::move(up_dec), hdr.source, hdr.destination);

        // 2. Classify and get parser
        mitvane::Classifier classifier(detection_context);
        ApplicationParser* parser = classifier.classify(m_app_layer_parsers);

        // 3. Apply application layer parser
        parser->parse(std::unique_ptr<vanetza::PacketVariant>(std::move(
            detection_context.payload)));

        // 4. Detect using decoded data, the current position and signatures
        // Detection for Geonetworking protocol
        if (signatures_.count(mitvane::Protocol::GeoNetworking)) {
            mitvane::GeonetDetector detector = mitvane::GeonetDetector(detection_context
                .geonet_data, positioning_);
            detector.detect(signatures_);
        }
        // Detection for
        // Application layer Detection

        //5. Forward
        destination_link_->transmit(hdr, packet);

    }

}
```

Listing 7.5. IDPS context

The code for `RuleReader`, `Decoder`, `Classifier`, `ApplicationParser` and `Detector` are available at the repository: `https://github.com/camsenec/Mitvane`. Besides, several significant codes are are shown in Appendix 1, including `RuleReader` (Listing 1, Listing 2), `Decoder` for GeoNetworking packet (Listing 3, Listing 4), `Classifier` (Listing 5, Listing 6) , `Detector` for GeoNetworking protocol (Listing 7, Listing 8) and `MessageHandler` (Listing 9, Listing 10)

# 8. Evaluation

The developed IDS/IPS is evaluated by security evaluation and performance. In the security evaluation, it is tested that risks shown in Table 12 are mitigated by the developed IDS/IPS. In the performance evaluation, the average time to process one packet is measured. With the measurement, the capability of the Mitvane to process packets can be determined.

## 8.1 Experimental setup

Experiments are conducted not using real-vehicle because safety is not guaranteed during the testing. This is one of the ethical issues considered in this research. For both security evaluation and performance evaluation, we need a targeted vehicle, where an OBU is installed. Besides, we need an attacker who has one or multiple OBUs and a program to generate malicious V2X messages. The OBU of the targeted vehicle is realized as a virtual machine on the VirtualBox platform.

On the virtual machine for the targeted vehicle, a program which receives V2X messages running. The program is Vanetza [55]-based ITS application (`https://github.com/camsenec/its_apps`). The ITS application verifies messages and transform the messages so that autonomous vehicle core process can use the messages. In Figure 22, the ITS application corresponds to the "ITS application", while Mitvane corresponds to "IDS/IPS".

The specs of the virtual machine are shown in Table 26. The values are determined based on the specs of Cohda MK5 OBU shown in Table 2. It is assumed that the accepted messages at the virtual machine are transformed and sent to the core process of self-driving, which runs on another host. However, in this experiment, the core process of self-driving is not executed because only whether the OBU accepts malicious messages is tested.

| **Name** | **Value** |
|---|---|
| CPU | AMD Ryzen 7 PRO 4750U with Radeon Graphics 1.7Ghz 1 Core 1 Thread |
| Memory | 1024 MByte |
| Operating System | Ubuntu 20.04 |
| Bandwidth | 10 MHz |

Table 26. *OBU of the targeted vehicle specs.*

The OBUs of the attacker are also realized by virtual machines. The virtual machines have the same spec as the targeted vehicle shown in Table 26. The messages for tests are sent from the attacker and received by the targeted vehicle. The network architecture for the experiment, which consists of the target vehicle's OBU, the attacker's OBUs is shown in Figure 27. OBUs are linked with each other through a virtual switch. Each OBUs are physically one-hop reachable by setting destination MAC as `ff:ff:ff:ff:ff:ff` or the destination's MAC address. Therefore, an ITS-S which is only reachable by setting a large number to maximum hop limit is not realized. This is one of the limitations of this experiment, but this problem is addressed in the next section.

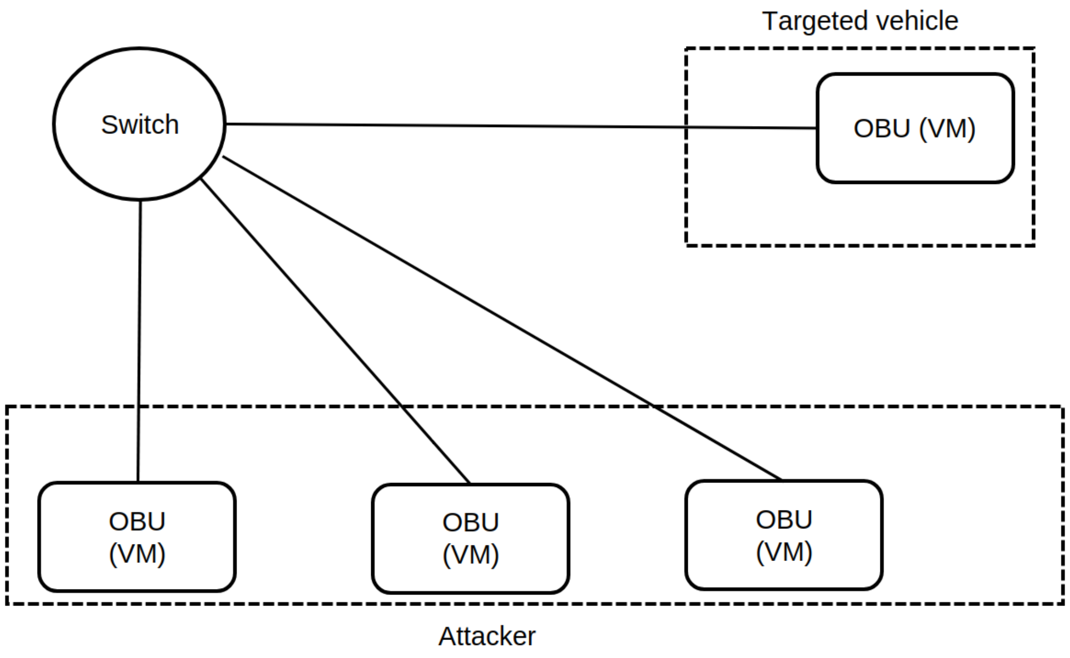


Figure 27. *Network architecture for the experiment.*

## 8.2 Security evaluation

### 8.2.1 Test case

The threats related to risks with HIGH or VERY HIGH value, AVs to cause the threats and countermeasures are shown in Table 27.

The test is conducted both for the case when the proposed IDS/IPS is enabled and when the IDS/IPS is disabled. The test cases are shown in Table 28. ID means test case ID, the threat is one of the threats that a threat actor tries to cause, the message is the content of malicious messages sent to a targeted vehicle to cause the threat, and attributes are values of specific fields to exploit vulnerabilities.

| **Threat** | **Attack vectors** | **Countermeasures** |
|---|---|---|
| Notification of false information | Obtaining the private key and the authorization ticket of an ITS-S (AV3), Single-Hop Broadcast of a message to ITS-Ss so that an unauthorized message is accepted (AV4), GeoBroadcast of a message to an targeted area with large maximum hop limit (AV5) | Blocking unauthorized messages, Blocking GBC packets with malicious configuration |
| Map database poisoning | Obtaining the private key and the authorization ticket of an ITS-S (AV3), Single-Hop Broadcast of a message to ITS-Ss so that an unauthorized message is accepted (AV4), GeoBroadcast of a message to an targeted area with large maximum hop limit (AV5) | Blocking unauthorized messages, Blocking GBC packets with malicious configuration |
| Tampering | Obtaining the private key and the authorization ticket of an ITS-S (AV3), Retransmission of a message to ITS-Ss which have been set as the destinations after capturing the message and modifying the content (AV6) | Blocking unauthorized messages, Blocking GBC packets with malicious configuration) |
| DoS | AV2 (False signature flood: send a V2X message with false signature many times) | Only allowing messages coming from trusted ITS-Ss |

Table 27. *Mapping of threats related to risks with HIGH value, AVs to cause the threats and countermeasures.*

In test case 1 and 2, CAMs, which includes information about a nonexistent vehicle in the destination area to cause notification of false information, are sent. Map database poisoning and tampering can be tested by the same test data as the data for notification of false information. In the map database poisoning, the message that should be sent is wrong static information such as wrong MAPEMs. The developed IDS/IPS blocks/allows messages based on the patterns in the GeoNetworking packet header and the ITS PDU header. Even if the message is changed to MAPEMs, the process for malicious packet detection does not change. Therefore, the test cases for notification of false information, map database poisoning is summed up into two test cases.

Also, regarding tampering, the message should be modified to include wrong information to cause an autonomous vehicle's misbehavior. Therefore, test cases 1 and 2 can also be tests for tampering. However, the destination and forwarding scheme is not changed in the attack chain for tampering. Therefore, tampered messages could be sent by SHB /TSB as well as GBC. Test case 3 tests the case when a message including wrong information is sent by SHB. In test cases 1 and 3, it is assumed that a private key and a valid authorization ticket have been obtained.

In test cases 1 and 3, the transmitted message is signed by the obtained private key. Besides, in test case 1, the destination area is set to a targeted area on top of GBC (AV5). The maximum hop limit is set to a large number to make the malformed message reach the destination area (AV4). The included false information is the one relevant to the targeted area. In test cases 2 and 3, the destination area is not explicitly configured. However, the vehicles to which messages are delivered are the ones reachable by one-hop. In that sense, the included information in test cases 2 and 3 is the one relevant to the nearby area for the threat actor.

For false signature-based DoS, messages with a false signature are sent with high frequency (AV2). Besides, the messages are sent from multiple OBUs. In test case 4, CAMs that include correct information are sent with 1 Hz frequency (1000 messages per second). The messages are transmitted by SHB. The message is sent from 3 OBUs.

### 8.2.2 Situation

The situation assumed in this experiment is shown in Figure 28. The attacker exists at $(latitude, longitude) = (48.7668616, 11.432068)$, which is the default value in Vanetza [55], when GPS position is configured as static. The targeted vehicle, which could be a victim of all threats shown in Table 27, exists at a random point in the square, where the length of all edges is 2000 m (2 km). The edges are perpendicular to the line of latitude

| Test case ID | Threat | Message | Attributes |
|---|---|---|---|
| 1 | Notification of false information (Map database poisoning, Tampering) | CAMs which includes information about a nonexistent vehicle in the destination area | authorized, GBC, estination area is a rectangle, where the length of the short edge (Distance $a$) is 200 m and the length of the long edge (Distance $b$) is also 200 m, 100 maximum hop limit |
| 2 | Notification of false information (Map database poisoning) | CAMs which includes information about a nonexistent vehicle in the destination area | unauthorized, SHB |
| 3 | Tampering | CAMs which includes information about a nonexistent vehicle in the destination area | authorized, SHB |
| 4 | DoS | CAMs | include a false signature, SHB, 1Hz frequency (1000 messages/s), 3 OBUs |

Table 28. *Test cases for notificaiton of false information, map database poisoning and deception.*

or the line of longitude.

The position of the target vehicle distiributes with the same probability in the square. It is known that given a base point represented by latitude ($lat1$) and longitude ($lon1$), azimuth (the heading measured clockwise from north) from the base point ($azi1$), and distance from the base point ($s12$), another point is determined uniquely [69]. Using this fact, the targeted vehicle's position, represented by $lat2$ and $lon2$ is determined by the algorithm shown as Algorithm 1, where the base point is the position of the attacker (*i.e.* $(lat1, lon1) = (48.766816, 11.432068)$). The $rand()$ function in Algorithm 1 returns pseudo-random number in the range between 0 and 1. The $direct()$ function is a function provided by GeoGraphicLib [70], which calculates $lat2$ or $lon2$ given $lat1$, $lon1$, $azi1$ and $s12$. In Algorithm 1, $lat2$ is firstly calculated by moving $s12\_north$ m toward north

**Algorithm 1** *Determination of the targeted vehicle's position.*

```
Require: $lat1, lon1$
1: $s12\_north \leftarrow rand() \times 2000 - 1000$
2: $lat2 \leftarrow direct(lat1, lon1, 0, s12\_north)$
3: $s12\_east \leftarrow rand() \times 2000 - 1000$
4: $lon2 \leftarrow direct(lat1, lon1, 90, s12\_east)$
```

($azi1 = 0$) from the base point. Next, $lon2$ is calculated by moving $s12\_east$ m toward

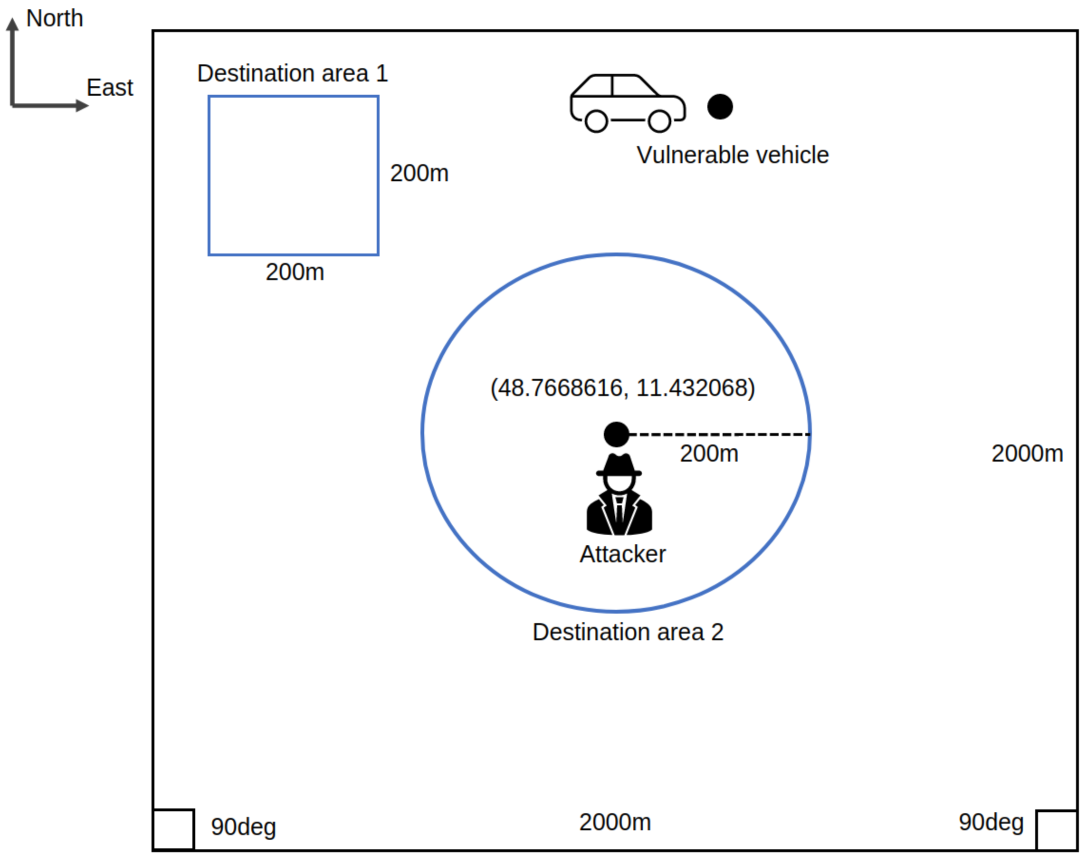


Figure 28. *Situation assumed in the experiment.*

| Area ID | Destination area | Test case IDs |
|---|---|---|
| 1 | A rectangle, where the length of the short edge is 200 m and the length of the long edge is also 200 m (*i.e.* a square, where the length of all edges are 200 m.) | 1 |
| 2 | A circle with radius 200 m | 2, 3 |

Table 29. *Destination areas used in the test cases.*

east ($asi1 = 90$) from the base point. Finally, the targeted vehicle's position is determined by $lat2$ and $lon2$.

The destination areas, which can be set in test case 1-3, are mapped to test case IDs as shown in Table 29. These destination areas are depicted in Figure 28. The first destination area is derived from test case 1. The second destination area is derived from the test cases 2 and 3. The destination area 2, a circle with a radius 200 m, is an implicit destination area determined by the range reachable by 1 hop. According to the specification of DSRC, which is an amendment of IEEE802.11p, the range reachable by 1 hop is 1 km [71]. However, this is the reachable range in theory. For that reason, the destination area when SHB is assumed as the circle with a radius 200 m, 20% of the theoretical distance. The center point of destination area 1 can be chosen by the attacker. Therefore, the attacker chooses the area where the target vehicle exists for the destination area. The center point of destination area 2 corresponds to the attacker's position.

It is assumed that the messages cannot be delivered from the attacker to vehicles outside of the destination area regardless of the fact that OBUs are physically reachable by one-hop in the network architecture shown in Figure 27. Also, it is assumed that, once the targeted vehicle is in the destination area, the message is always delivered to the targeted vehicle from the attacker by hops. In test case 4, it is assumed that the targeted vehicle is always reachable from the attacker.

### 8.2.3 Metrics and methodology

The result of tests are given by SUCCESS or FAIL. SUCCESS means that the malformed messages are blocked. FAIL means that the malformed messages are accepted, which means that the malformed messages cannot be blocked. Between test cases 1-3 and test case 4, different metrics are used.

From here, the metric for the test cases 1-3 is described. The probability of a message being reached to the targeted vehicle is summarized in Table 30, where each area in Table 29 is represented by the ID. The probability of a message being reached to the

| **Destination area** | **Probability ($\mu_0$)** |
|---|---|
| 1 | $1.0000$ |
| 2 | $0.0314$ |

Table 30. *Probability of a message being reached to the targeted vehicle.*

targeted vehicle is mathematically calculated because it is equivalent to the probability that the targeted vehicle is within the destination area. For destination area 1, this probability is always 1 because an attacker can choose the destination area freely. On the other hand, the probability that the targeted vehicle is within the destination area for destination area 2 is calculated by the size of a destination area divided by the size of the square where the targeted vehicle can exist. The probability of a message being reached to the targeted vehicle for area 2 is calculated as follows.

1. the size of the destination area: $40000\pi$ m$^2$
2. the size the square where the targeted vehicle can exist: $4000000$ m$^2$
3. probability of a message being reached to the targeted vehicle: $\frac{40000\pi}{4000000} \approx 0.0314$

If the probability of a message being accepted is not different from the probability shown in Table 30 so much, it can be said that the malformed messages could not be blocked (FAIL). In order to evaluate the difference mathematically, hypothesis testing is adopted. The Hypothesis test involves making a null hypothesis, which should be rejected, and collecting data from an experiment [72]. Then, a decision can be made as to whether or not there is sufficient evidence to reject the null hypothesis based on analyses of the data.

Denoting the population mean of messages being accepted by the targeted vehicle $\mu$, the null hypothesis ($H_0$) and alternative hypothesis ($H_1$) are defined as follows:

- $H_0 : \mu = \mu_0$,
- $H_1 : \mu \neq \mu_0$,

, where $\mu_0$ is a probability mathmatically calculated as shown in Table 30. If the null hypothesis $H_0$ is rejected, it can be said that the malformed messages are blocked (SUCCESS).

The methodology to do the experiment is described in the following First, $1000$ malformed messages are sent while changing the position of the targeted vehicle to measure the probability of a malformed message being accepted by the target vehicle. The method to choose the position of the targeted vehicle is described in Section 8.2.2. Then, a message acceptance rate can be calculated. Second, this $1000$ transmission of messages is repeated

100 times. After the experiment, the mean of the message acceptance rate can be calculated from the 100 samples. Besides, the standard deviation can be calculated from the 100 samples. The mean is denoted by $\bar{x}$, while the standard deviation is denoted by $s$.

Then, using the value of $\bar{x}$, $\mu_0$, $s$ and the sample size $n = 100$, a score called $t$-score can be calculated. By the $t$-score, it is determined whether the estimated population mean of the message acceptance rates can is close enough to $\mu_0$. The $t$-score can be calculated by the following equation [72].

$$t = \begin{cases} \frac{\bar{x}-\mu_0}{\sqrt{\frac{s^2}{n}}} & (\bar{x} \neq \mu_0) \\ 0 & (\bar{x} = \mu_0) \end{cases} \tag{8.1}$$

The $t$-scores follow a $t$-distribution with $n - 1$ degrees of freedom.

In this experiment, the sample size is 100. Therefore, the degrees of freedom is 99. The range $t$-score should be exists can be determined using $t$-distribution table [73] and a value called $p$-value. For instance, when $p$-value is 0.05, the $\mu_0$ is in the 95%confidential interval of the distribution of the message acceptance rate observed in the experiment. In this thesis, $p = 0.05$ is adopted. Then, if the $t$-score is not in the range 8.2, the null hypothesis $H_0$ is rejected.

$$t < -1.984 \lor t > 1.984 \tag{8.2}$$

It is defined that when the $H_0$ is rejected, the test result is SUCCESS (*i.e.* the malformed messages were blocked).It is defined that when the $H_0$ is not rejected, the test result is FAIL (*i.e.* the malformed messages were not blocked). On the other hand, it is defined that when the $H_0$ is rejected, the test result is SUCCESS (*i.e.* the malformed messages were blocked). he probability of a message being accepted will also be shown as results.

For test case 4, simple criteria are adopted. When over 10% of CAMs cannot be processed within 2 seconds after the attacker send the messages, the test result is FAIL, otherwise SUCCESS. The value 2 seconds comes from C2C-CC Basic System Profile v1.1.0 by CAR2CAR communication consortium [54]. The system profile states that the receiver should accept CAMs created in the last 2 seconds. This means that a CAM received more than 2 seconds after it was generated is useless, which results in loss of availability of the CAM.

### 8.2.4 Test result

The simulation is conducted following the situatoin and methodology described in the previous sections. The rule file used for this simulation is shown in Listing 8.1. Signature 1 blocks packets without the secure header in the GeoNetworking packet. Signature 2 blocks packets with the maximum hop limit larger than 3. Signature 3 blocks packets from out of a circle with a radius 100 m centered on the ego-vehicle. Signature 4 blocks packets that come from ITS-Ss whose IDs are not listed in `allowed_ids`. With the detection file, Mitvane is installed on the target vehicle.

All of the attacker's OBU is not the ones in the `allowed_ids` of signature 4. Signature 4 is an effective method to block packets, but it is too strict. All packets from the ITS-Ss with IDs which is not included in the list are broked. This strict blocking could reduce the number of correct messages that the vehicle with Mitvane can receive. Therefore, the test for the case when methane is enabled is divided into two tests, namely, when signature 4 is enabled (strict) and when signature 4 is not enabled (not strict).

```yaml
---
# action: "drop", "alert"
# protocol: "GeoNetworking", "BTP", "Facility", "CA", "DEN", "RLT", "TLM"

rules:
  # Signature 1 (FR1)
  - action: "drop"
    protocol: "GeoNetworking"
    meta:
      msg: "Unauthorized message is detected."
      sid: 1
      rev: 1
    nh: "common"

  # Signature 2 (FR2 (maximum hop limit))
  - action: "drop"
    protocol: "GeoNetworking"
    meta:
      msg: "GBC with large max hop limit is detected."
      sid: 2
      rev: 1
    ht: "GBC"
    mhl: 3

  # Signature 3 (FR2 (destination area))
  - action: "drop"
    protocol: "GeoNetworking"
    meta:
      msg: "GBC from a source far from the ego-vehicle is detected"
      sid: 3
      rev: 1
    ht: "GBC"
    allowed_so_range:
      shape: "circle"
```

| **Test case ID** | **Threat** | **Mean** | **Std** | ***t*-score** | **Result** |
|---|---|---|---|---|---|
| 1 | Notification of false information (Map database poisoning, Tampering) | 1.0000 | 0.0000 | 0.0000 | FAIL |
| 2 | Notification of false information (Map database poisoning) | 0.03201 | 0.0054 | 1.1325 | FAIL |
| 3 | Tampering | 0.03151 | 0.0057 | 0.1913 | FAIL |
| 4 | DoS | 0.1318 | 0.0461 |  | FAIL |

Table 31. *Test results (When Mitvane is disabled).*

```
        distance_a: 100
        distance_b: 100

  # Signature 4 (FR3 (Identity based))
  - action: "drop"
    protocol: "facility"
    meta:
      msg: "A packet from not trusted ITS-S is detected. (Identity based)"
      sid: 4
      rev: 1
    allowed_ids:
      - 123490
      - 123491
      - 123492
```

Listing 8.1. Detection rule for evaluation

The test results when Mitvane is disabled are shown in Table 31. Besides, the test results when Mitvane is enabled (signature 4 is not enabled) is shown in Table 32, while the test results when Mitvane is enabled (signature 4 is enabled) is shown in Table 33. For test cases 1-3, the mean represents the average probability that messages are accepted, while the std represents the standard variation of the probability that messages are accepted. For test case 4, the mean represents the average probability that messages are processed within 2 seconds since the attacker generates the messages, while the std represents the standard variation of the probability that messages are processed within 2 seconds since the attacker generates the messages. The result is given by SUCCESS or FAIL.

## 8.2.5 Discussion

This section discusses the test results and describes which signature in Listing 8.1 blocked packets when the test result is SUCCESS. Firstly, the results for test cases 1-3 are discussed. When Mitvane is not enabled, the average probability that messages are accepted is not

| **Test case ID** | **Threat** | **Mean** | **Std** | $t$**-score** | **Result** |
|---|---|---|---|---|---|
| 1 | Notification of false information (Map database poisoning, Tampering) | 0.0000 | 0.0000 | $\infty$ | SUCCESS |
| 2 | Notification of false information (Map database poisoning) | 0.0000 | 0.0000 | $\infty$ | SUCCESS |
| 3 | Tampering | 0.00756 | 0.0026 | -3760.11 | SUCCESS |
| 4 | DoS | 0.1628 | 0.0052 | | FAIL |

Table 32. *Test results (When Mitvane is enabled and Signature 4 is not enabled).*

| **Test case ID** | **Threat** | **Mean** | **Std.** | $t$**-score** | **Result** |
|---|---|---|---|---|---|
| 1 | Notification of false information (Map database poisoning, Tampering) | 0.0000 | 0.0000 | $\infty$ | SUCCESS |
| 2 | Notification of false information (Map database poisoning) | 0.0000 | 0.0000 | $\infty$ | SUCCESS |
| 3 | Tampering | 0.0000 | 0.0000 | $\infty$ | SUCCESS |
| 4 | DoS | 1.000 | 0.0000 | | SUCCESS |

Table 33. *Test results (When Mitvane is enabled and Signature 4 is enabled).*

| **Test case ID** | **Signatures** |
|---|---|
| 1 | Signature 2, Signature 3 |
| 2 | Signature 1 |
| 3 | Signature 3 |
| 4 | |

Table 34. *Signatures that matches with the received packets (when signature 4 is not enabled).*

| **Test case ID** | **Signatures** |
|---|---|
| 1 | Signature 2, Signature 3, Signature 4 |
| 2 | Signature 1, Signature 4 |
| 3 | Signature 3, Signature 4 |
| 4 | Signature 4 |

Table 35. *Signatures that matches with the received packets (when signature 4 is enabled).*

different so much from the probability that the messages are reached to the target vehicle, as the $t$-score shows. On the other hand, when Mitvane is enabled, the mean message acceptance rate is decreased for test cases 1, 2, and 3. The test results for these test cases become SUCCESS following the metric 8.2. However, the mean malicious message acceptance rate does not become $0$ in test case 3 when signature 4 is not enabled. Against the false signature-based DoS, Mitvane cannot block packets with false signatures without Signaure 4. However, when signature 4 is enabled, the average probability that messages are processed within 2 seconds since the attacker generates the messages become $1.000$.

Which signatures in Listing 8.1 matches patterns in the received packets is described in table 34 (when signature 4 is not enabled) and in Table 35 (when signature 4 is enabled) to analyze the cause why the mean malicious message acceptance rate does not become $0$ in Table 32 for several test cases.

From Table 34, the signature whose action is performed in test case 3 is signature 3. Signature 3 blocks packets that come from out of a circle with a radius $100$ m centered on the target vehicle. Therefore, it can be inferred that when the distance between the attacker and the target vehicle is less than $100$ m, the packets are not blocked. The packets with a large maximum hop limit are blocked by signature 2, while the packets coming from far from the target vehicle are blocked by signature 3.

From Table 35, signature 4 is effective to prevent false signature flood. Mitvane does not do message verification or duplicate packet detection. Because of the lack of cryptographic message verification, the average time for Mitvane to process one packet is less than $1$ ms, while the average time that vanetza-based ITS-application, which verifies and transform messages, processes one packet is $8.250$ ms. By blocking packets from basic information included in the packets by Mitvane, non-necessary verification process can be skipped. And then, the probability of DoS is reduced.

## 8.3 Performance evaluation

In the performance evaluation, a malformed packet used in the test case 1 of the security evaluation is sent from the attacker to the target vehicle. Then, the time to inspect the packet, detect a malicious pattern and handle the packet is measured. The number of packets sent from the attacker is $1000$.

Figure 29 and Table 36 shows the result of performance evaluation. In addition to the total processing time by Mitvane from packet acquisition to message forwarding, the processing time by each component in Figure 26, namely decoder, classifier, application layer parser, detector and message handler is measured. Note that total processing time does not correspond to the sum of processing time by each component because the total processing time includes the time for packet aquisition and forwarding.

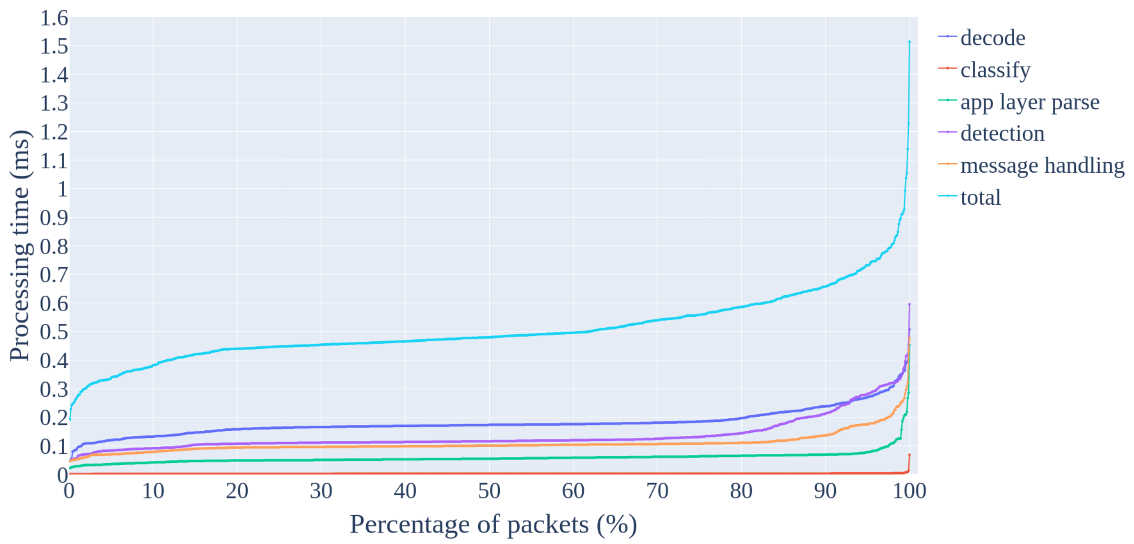


Figure 29. *The time to inspect the packet, detect a malicious pattern and handle the packet is measured against percentage of packets.*

Figure 29 shows how many percentages of packets are processed within a processing time. For example. the total processing time is less than $0.6$ ms for $90\%$ of packets. From the graph, it can be observed that over $90\%$ of the packets are processed within no more than $1.6$ ms. The most time-consuming components are decoder, detector and message handler.

| Component | Average (ms) | Std. (ms) | Max (ms) | Min (ms) |
|---|---|---|---|---|
| decoder | 0.181 | 0.046 | 0.508 | 0.047 |
| classifier | 0.003 | 0.002 | 0.069 | 0.001 |
| app layer parser | 0.059 | 0.024 | 0.453 | 0.022 |
| detector | 0.136 | 0.060 | 0.596 | 0.049 |
| message handler | 0.108 | 0.036 | 0.477 | 0.048 |
| total | 0.508 | 0.122 | 1.514 | 0.193 |

Table 36. *Average, standard deviation, max and min of the processing time by each component and total processing time.*

However, the procesing time for one component is less than $0.6$ ms. Table 36 shows average, standard deviation, max and min of the processing time by each component and total processing time. From the table, the average of total processing time is $0.508$ ms. This implies that Mitvane can handle around 2000 messages per second (2000 Hz). Considering the most frequently received V2X message, CAM, is received with 1 $Hz$ to 10 $Hz$ [44], it can be said that Mitvane is capable of processing V2X messages exchanged in ETSI ITS-G5 V2X communication system.

## 8.4 Potential bypass of Mitvane

Finally, whether the attackers can bypass the developed IDS/IPS, Mitvane, or induce DoS despite the existence of the Mitvane. First, whether the attackers can bypass Mitvane is discussed for each signature in 8.1.

It has been verified that Signature 1 can block unauthorized packets by checking the NH field of the GeoNetworking packet. As described in Section 6.2, the NH field of the basic header is set to the common header if a certificate is not included, while the field is set to the secure header otherwise. Here, attackers can set the NH field to a secure header but do not include a certificate. This enables attackers to make the unauthorized messages ignored by the IDS/IPS. The solution is to check the existence of a certificate as well as the NH field. Then, even when the NH field is forged, the IDS/IPS can check whether the message is authorized or not.

For the signatures to block TSB/GBC packets with malicious configuration (Signature 2, Signature 3), attackers cannot bypass Mitvane because changing the forwarding scheme, destination area or maximum hop limit also changes the forwarding way of the packets. If an attacker sets a legitimate value to the TSB/GBC relevant field, the attacker cannot perform the intended attack.

For the signature to only allow messages from trusted ITS-Ss (signature 4), attackers can bypass the Mitvane by forging the station ID. If an attacker obtains a station ID of a trusted vehicle listed in the detection rule and sets the station ID to the header of V2X messages, the V2X messages would be accepted. The solution is to conduct a certificate-based check, not the ID-based check, where trusted certificates are listed and stored in advance along with Mitvane. Then, if the certificate included in a packet corresponds to one of the trusted certificates, it can be said that the packet comes from trusted ITS-Ss.

Performance evaluation reveals that the Mitvane is capable of processing around 2000 messages per second. However, if IDS/IPS receives over 2000 messages per second, the IDS/IPS would lose availability. This is a limitation of the IDS/IPS. A solution to mitigate this issue is to deploy multiple IDSs/IPSs. In general, a network interface can be bound to multiple processes. Therefore, packets can be distributed to multiple Mitvanes. Theoretically, by adding one IDS/IPS, the number of messages which can be processed in one-second increases by 2000.

However, in the first place, the ITS application, which verifies and passes messages to the autonomous vehicle core process, cannot process 2000 messages in one second. The average message verification and transformation time for ITS application is around 8.250 ms as described in Section 8.2.5. Therefore, even if the Mitvane can process over 2000 packets, the ITS application cannot handle so many messages.

The benefit of deploying IDS/IPS to prevent DoS attacks is to skip unnecessary verification and transformation for the packets coming from an attacker conducting a DoS attack. Because Mitvane can process one packet within 1 ms on average, Mitvane would help ITS application focus on verifying and transforming only potentially legitimate packets.

# 9. Summary

This thesis analyzed the cybersecurity risks of utilizing messages coming from external vehicles/infrastructures in the ETSI ITS-G5 V2X communication system to control vehicles autonomously. The analysis was conducted following PASTA (Process for Attack Simulation and Threat Analysis). In the risk analysis, enumeration of threats against autonomous vehicles, which uses messages coming from external vehicles/infrastructures, and analysis of the unique weaknesses/vulnerabilities of the ETSI ITS-G5 V2X communication system were conducted. Based on the identification of the weaknesses and vulnerabilities, several attack scenarios with the goals – to cause misbehavior of an autonomous vehicle and to prevent an autonomous vehicle from receiving messages, were described in detail. Attack vectors were identified after organizing attack scenarios into attack trees. Finally, the cybersecurity risks of utilizing messages from other vehicles/infrastructures on the ETSI ITS-G5 V2X communication system to autonomously control vehicles are identified. The identified risks are shown in Table 13. Besides, countermeasures to mitigate the identified risks are proposed. The countermeasures for each identified risk is summarized in Table 13.

The packets which should be blocked as the countermeasures can be identified from the information included in the GeoNetworking packet, which is the specification of the packet exchanged in ETSI ITS-G5-based V2X communication. Therefore, as the combination of the countermeasures, signature-based IDS/IPS named Mitvane was proposed. The functional requirements of the IDS/IPS were determined to achieve the countermeasures and mitigate the identified risks. The functional requirements (FRs) are listed in Chapter 6, while the methods to be implemented in Mitvane to satisfy the functional requirements is shown in Table 24. Each method is represented by a detection rule in Listing 7.1. Then, Mitvane detects maliciously configured packets and messages based on the detection rules. Mitvane was developed in object-oriented design so that additional detection modules can be integrated easily. Finally, it was verified that Mitvane could mitigate the identified risks of utilizing messages from other vehicles/infrastructures on the ETSI ITS-G5 V2X communication system to autonomously control vehicles.

The answers for the research questions of this thesis is summarized in Table 37.

| RQ | Question | Answer |
|---|---|---|
| *RQ-1* | What are the cybersecurity risks of utilize messages coming from other vehicles/infrastructures on the ETSI ITS-G5 V2X communication system to control vehicles autonomously? | The four risks are shown in Table 13. |
| *RQ-2* | What countermeasures are effective to mitigate the identified cybersecurity risks given as the answer of *RQ-1*? | The countermeasures for each risk are shown in Table 13. |
| *RQ-3* | What functions are required for IDS/IPS to realize countermeasures given as the answer of *RQ-2*? | The functions are<br>■ The IDS/IPS can block unauthorized messages,<br>■ The IDS/IPS can block TSB/GBC packets with malicious configuration,<br>■ The IDS IPS can only allows messages coming from trusted ITS-Ss,<br>as described in Chapter 6. The methods to achieve these functions are shown in Table 24. |
| *RQ-4* | How the IDS/IPS should be implemented to satisfy the requirements given as the answer of *RQ-3*? | The IDS/IPS is developed to detect malicious packets and messages based on detection rules. Each detection rule represents patterns that should be detected and blocked to satisfy functional requirements. |

Table 37. *Research questions and answers.*

## 9.1 Future work

In this thesis, the IDS/IPS was developed in the identified risk-oriented way, which means that the IDS/IPS was developed specifically to mitigate identified risks in the risk analysis. However, Mitvane has the potential to perform various detection of packets exchanged in ETSI iTS-G5-based V2X communication by supporting additional keywords. For example, the position vector of the GeoNetworking packet includes TST field, which represents the time when the latitude and longitude of the node are acquired, as shown in Table 17. This field was not used to realize the proposed countermeasures. However, the time when the geographic position is acquired is important information to check whether the included source position is too old or not. If the time is too old, the source position cannot be trusted because the sender vehicle would have moved to another place. Including this example, one of the future works is to improve the IDS/IPS so that users can specify various keywords and block/allow packets following their own risk analysis and countermeasures.

Another future work is to improve the performance of the IDS/IPS. One of the way to improve performance is threading. For example, in the detection module, the detections based on signatures can be divided into small tasks. Then, each thread can take responsibility for several small tasks. Besides, the solution to mitigate potential bypass of the proposed IDS/IPS discussed in Section 8.4 will be implemented.

# Bibliography

[1] Sparsh Sharma and Ajay Kaul. “A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud”. In: *Vehicular Communications* 12 (2018), pp. 138–164. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2018.04.005. URL: https://www.sciencedirect.com/science/article/pii/S2214209617302784.

[2] “IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Networking Services”. In: *IEEE Std 1609.3-2020 (Revision of IEEE Std 1609.3-2016)* (2021), pp. 1–210. DOI: 10.1109/IEEESTD.2021.9374154.

[3] *Suricata*. URL: https://suricata.io/ (visited on 05/13/2022).

[4] Arooj Masood, Demeke Shumeye Lakew, and Sungrae Cho. “Security and Privacy Challenges in Connected Vehicular Cloud Computing”. In: *IEEE Communications Surveys Tutorials* 22.4 (2020), pp. 2725–2764. DOI: 10.1109/COMST.2020.3012961.

[5] Tao Zhang and Luca Delgrossi. *Vehicle safety communications: protocols, security, and privacy*. Vol. 103. John Wiley & Sons, 2012.

[6] Georgios Karagiannis et al. “Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions”. In: *IEEE Communications Surveys Tutorials* 13.4 (2011), pp. 584–616. DOI: 10.1109/SURV.2011.061411.00019.

[7] ETSI. *Intelligent Transport Systems (ITS); Communications Architecture*. Tech. rep. ETSI EN 302 665 (v1.1.1). ETSI, Sept. 2010.

[8] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements*. Tech. rep. ETSI TS 102 637-1, (v1.1.1). ETSI, Sept. 2010.

[9] SAE. *Surface vehicle standard: Dedicated Short Range Communications (DSRC) Message Set Dictionary*. Tech. rep. J2735. SAE International, Mar. 2016.

[10] Norbert Varga et al. “An architecture proposal for V2X communication-centric traffic light controller systems”. In: *2017 15th International Conference on ITS Telecommunications (ITST)*. 2017, pp. 1–7. DOI: 10.1109/ITST.2017.7972217.

[11] Manabu Tsukada et al. "AutoC2X: Open-source software to realize V2X cooperative perception among autonomous vehicles". In: *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. 2020, pp. 1–6. DOI: 10.1109/VTC2020-Fall49728.2020.9348525.

[12] Mai Hirata et al. "Roadside-assisted Cooperative Planning using Future Path Sharing for Autonomous Driving". In: (2021). arXiv: 2108.04629. URL: http://arxiv.org/abs/2108.04629.

[13] Mao Shan et al. "Demonstrations of Cooperative Perception: Safety and Robustness in Connected and Automated Vehicle Operations". In: *Sensors* 21.1 (2021). ISSN: 1424-8220. DOI: 10.3390/s21010200. URL: https://www.mdpi.com/1424-8220/21/1/200.

[14] ETSI. *Intelligent Transport Systems (ITS); Security; Security header and certificate formats*. Tech. rep. ETSI TS 103 097 (v1.4.1). ETSI, Oct. 2020.

[15] Tiffany Hyun-Jin Kim et al. "VANET Alert Endorsement Using Multi-Source Filters". In: *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*. VANET '10. Chicago, Illinois, USA: Association for Computing Machinery, 2010, pp. 51–60. ISBN: 9781450301459. DOI: 10.1145/1860058.1860067. URL: https://doi.org/10.1145/1860058.1860067.

[16] M. Raya et al. "Eviction of misbehaving and faulty nodes in vehicular networks". In: *IEEE Journal on Selected Areas in Communications* 25.8 (2007). cited By 315, pp. 1557–1568. DOI: 10.1109/JSAC.2007.071006. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-35348977820&doi=10.1109%2fJSAC.2007.071006&partnerID=40&md5=6b438237b0c3a23a39d46ca30ae3771f.

[17] Zhen Cao et al. "Proof-of-relevance: Filtering false data via authentic consensus in Vehicle Ad-hoc Networks". In: *IEEE INFOCOM Workshops 2008*. 2008, pp. 1–6. DOI: 10.1109/INFOCOM.2008.4544650.

[18] Jonathan Petit, Michael Feiri, and Frank Kargl. "Spoofed data detection in VANETs using dynamic thresholds". In: *2011 IEEE Vehicular Networking Conference (VNC)*. 2011, pp. 25–32. DOI: 10.1109/VNC.2011.6117120.

[19] Irshad Ahmed Sumra et al. "Classes of attacks in VANET". In: *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*. 2011, pp. 1–5. DOI: 10.1109/SIECPC.2011.5876939.

[20] Joseph Soryal and Tarek Saadawi. "DoS attack detection in Internet-connected vehicles". In: *2013 International Conference on Connected Vehicles and Expo (ICCVE)*. 2013, pp. 7–13. DOI: 10.1109/ICCVE.2013.6799761.

[21] Chaker Abdelaziz Kerrache et al. "TFDD: A trust-based framework for reliable data delivery and DoS defense in VANETs". In: *Vehicular Communications* 9 (2017), pp. 254–267.

[22] Amrita Ghosal and Mauro Conti. "Security issues and challenges in V2X: A Survey". In: *Computer Networks* 169 (2020), p. 107093. ISSN: 1389-1286. DOI: `https://doi.org/10.1016/j.comnet.2019.107093`. URL: `https://www.sciencedirect.com/science/article/pii/S1389128619305857`.

[23] Ming-Chin Chuang and Jeng-Farn Lee. "TEAM: Trust-extended authentication mechanism for vehicular ad hoc networks". In: *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*. 2011, pp. 1758–1761. DOI: `10.1109/CECNET.2011.5768376`.

[24] Li He and Wen Tao Zhu. "Mitigating DoS attacks against signature-based authentication in VANETs". In: *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*. Vol. 3. 2012, pp. 261–265. DOI: `10.1109/CSAE.2012.6272951`.

[25] John R. Douceur. "The Sybil Attack". In: *Peer-to-Peer Systems*. Ed. by Peter Druschel, Frans Kaashoek, and Antony Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260. ISBN: 978-3-540-45748-0.

[26] Philippe Golle, Dan Greene, and Jessica Staddon. "Detecting and correcting malicious data in VANETs". In: *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. 2004, pp. 29–37.

[27] Bin Xiao, Bo Yu, and Chuanshan Gao. "Detection and Localization of Sybil Nodes in VANETs". In: *Proceedings of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*. DIWANS '06. Los Angeles, CA, USA: Association for Computing Machinery, 2006, pp. 1–8. ISBN: 1595934715. DOI: `10.1145/1160972.1160974`. URL: `https://doi.org/10.1145/1160972.1160974`.

[28] ByungKwan Lee, EunHee Jeong, and Ina Jung. "A DTSA (detection technique against a sybil attack) protocol using SKC (session key based certificate) on VANET". In: *International Journal of Security and Its Applications* 7.3 (2013), pp. 1–10.

[29] Mina Rahbari and Mohammad Ali Jabreil Jamali. "Efficient detection of Sybil attack based on cryptography in VANET". In: *arXiv preprint arXiv:1112.2257* (2011).

[30] Xia Feng et al. "A method for defensing against multi-source Sybil attacks in VANET". In: *Peer-to-Peer Networking and Applications* 10 (2017), pp. 305–314.

[31] ETSI. *Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management*. Tech. rep. ETSI TS 102 940 v1.3.1. ETSI, Apr. 2018.

[32] Hind Bangui and Barbora Buhnova. “Recent Advances in Machine-Learning Driven Intrusion Detection in Transportation: Survey”. In: *Procedia Computer Science* 184 (2021). The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops, pp. 877–886. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2021.04.014. URL: https://www.sciencedirect.com/science/article/pii/S1877050921007894.

[33] Hichem Sedjelmaci, Sidi Mohammed Senouci, and Mosa Ali Abu-Rgheff. “An Efficient and Lightweight Intrusion Detection Mechanism for Service-Oriented Vehicular Networks”. In: *IEEE Internet of Things Journal* 1.6 (2014), pp. 570–577. DOI: 10.1109/JIOT.2014.2366120.

[34] N. Venkatadri and K. Ramesh Reddy. “Secure TORA: Removal of Black Hole Attack Using Twofish Algorithm”. In: *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. 2016, pp. 239–244. DOI: 10.1109/IACC.2016.53.

[35] Sushmita Ruj et al. “On Data-Centric Misbehavior Detection in VANETs”. In: *2011 IEEE Vehicular Technology Conference (VTC Fall)*. 2011, pp. 1–5. DOI: 10.1109/VETECF.2011.6093096.

[36] Andreas Tomandl, Karl-Peter Fuchs, and Hannes Federrath. “REST-Net: A dynamic rule-based IDS for VANETs”. In: *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*. 2014, pp. 1–8. DOI: 10.1109/WMNC.2014.6878854.

[37] Norbert Bißmeyer, Christian Stresing, and Kpatcha M. Bayarou. “Intrusion detection in VANETs through verification of vehicle movement data”. In: *2010 IEEE Vehicular Networking Conference*. 2010, pp. 166–173. DOI: 10.1109/VNC.2010.5698232.

[38] Jay Rupareliya, Sunil Vithlani, and Chirag Gohel. “Securing VANET by Preventing Attacker Node Using Watchdog and Bayesian Network Theory”. In: *Procedia Computer Science* 79 (2016). Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016, pp. 649–656. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.03.082. URL: https://www.sciencedirect.com/science/article/pii/S1877050916002131.

[39] Uzma Khan, Shikha Agrawal, and Sanjay Silakari. "Detection of Malicious Nodes (DMN) in Vehicular Ad-Hoc Networks". In: *Procedia Computer Science* 46 (2015). Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India, pp. 965–972. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.01.006. URL: https://www.sciencedirect.com/science/article/pii/S1877050915000071.

[40] Omar Abdel Wahab et al. "CEAP: SVM-based intelligent detection model for clustered vehicular ad hoc networks". In: *Expert Systems with Applications* 50 (2016), pp. 40–54. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2015.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0957417415008088.

[41] K. Zaidi et al. "Host-Based Intrusion Detection for VANETs: A Statistical Approach to Rogue Node Detection". In: *IEEE Transactions on Vehicular Technology* 65.8 (2016). cited By 92, pp. 6703–6714. DOI: 10.1109/TVT.2015.2480244. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84982273121&doi=10.1109%2fTVT.2015.2480244&partnerID=40&md5=7b2de7d69bffc85eb93d27cf51b818a1.

[42] Min-Joo Kang and Je-Won Kang. "Intrusion detection system using deep neural network for in-vehicle network security". In: *PloS one* 11.6 (2016), e0155781.

[43] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications;Definitions*. Tech. rep. ETSI TR 102 638 (v1.1.1). ETSI, June 2009.

[44] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. Tech. rep. ETSI EN 302 637-2 (v1.4.1). ETSI, Apr. 2019.

[45] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. Tech. rep. ETSI EN 302 637-3 (v1.3.1). ETSI, Apr. 2019.

[46] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Facilities layer protocols and communication requirements for infrastructure services*. Tech. rep. ETSI TS 103 097 (v1.4.1). ETSI, Feb. 2020.

[47] Andreas Festag. "Standards for vehicular communication—from IEEE 802.11p to 5G". In: *e & i Elektrotechnik und Informationstechnik* 132 (Sept. 2015). DOI: 10.1007/s00502-015-0343-0.

[48] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*. Tech. rep. ETSI EN 302 636-1 v1.2.1. ETSI, Apr. 2014.

[49] ETSI. *Intelligent Transport Systems (ITS); Security; Trust and Privacy Management*. Tech. rep. ETSI TS 102 941 v1.4.1. ETSI, Jan. 2021.

[50] Shaoshan Liu et al. "Edge Computing for Autonomous Driving: Opportunities and Challenges". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1697–1716. DOI: 10.1109/JPROC.2019.2915983.

[51] Shaoshan Liu et al. *Creating Autonomous Vehicle Systems*. 2020.

[52] Cohda Wireless. *MK5 OBU Product Brief Sheet*. 2018. URL: https://www.cohdawireless.com/wp-content/uploads/2018/08/CW_Product-Brief-sheet-MK5-OBU.pdf (visited on 03/16/2022).

[53] ETSI. *Intelligent Transport Systems (ITS); Security; Security header and certificate formats*. Tech. rep. ETSI TS 103 097 (v1.2.1). ETSI, June 2015.

[54] CAR 2 CAR Communication Consortium. *C2C-CC Basic System Standards Profile*. Tech. rep. v1.1.0. CAR 2 CAR Communication Consortium, Dec. 2015.

[55] Raphael Riebl et al. "Vanetza: Boosting research on inter-vehicle communication". In: *Proceedings of the 5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2017)* (2017), pp. 37–40.

[56] Deborah J. Bodeau, Catherine D. McCollum, and David B. Fox. "Cyber Threat Modeling: Survey, Assessment, and Representative Framework". In: *Homeland Security Systems Engineering and Development Institute (HSSEDI)* (Apr. 2020).

[57] "Intro to Pasta". In: *Risk Centric Threat Modeling*. John Wiley & Sons, Ltd, 2015. Chap. 6, pp. 317–342. ISBN: 9781118988374. DOI: https://doi.org/10.1002/9781118988374.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118988374.ch6. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118988374.ch6.

[58] I. Tarandach and M.J. Coles. *Threat Modeling: A Practical Guide for Development Teams*. O'Reilly Media, Incorporated, 2020. ISBN: 9781492056553. URL: https://books.google.ee/books?id=WcbFyQEACAAJ.

[59] "Pasta Use Case". In: *Risk Centric Threat Modeling*. John Wiley & Sons, Ltd, 2015. Chap. 8, pp. 479–632. ISBN: 9781118988374. DOI: https://doi.org/10.1002/9781118988374.ch8. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118988374.ch8. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118988374.ch8.

[60] Jonathan Petit and Steven E. Shladover. “Potential Cyberattacks on Automated Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2015), pp. 546–556. DOI: `10.1109/TITS.2014.2342271`.

[61] Ahmed Shoeb Al Hasan, Md. Shohrab Hossain, and Mohammed Atiquzzaman. “Security threats in vehicular ad hoc networks”. In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2016, pp. 404–411. DOI: `10.1109/ICACCI.2016.7732079`.

[62] Mani Amoozadeh et al. “Security vulnerabilities of connected vehicle streams and their impact on cooperative driving”. In: *IEEE Communications Magazine* 53.6 (2015), pp. 126–132. DOI: `10.1109/MCOM.2015.7120028`.

[63] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality*. Tech. rep. ETSI EN 302 636-4-1 v1.4.1. ETSI, Jan. 2020.

[64] Bruce Schneier. *Attack Trees*. 1999. URL: `https://www.schneier.com/academic/archives/1999/12/attack_trees.html` (visited on 03/26/2022).

[65] Free Software Foundation. *GNU Lesser General Public License*. URL: `https://www.gnu.org/licenses/lgpl-3.0.en.html` (visited on 04/13/2022).

[66] Boost.org. *boost C++ libraries*. URL: `https://www.boost.org/` (visited on 04/13/2022).

[67] URL: `https://github.com/jbeder/yaml-cpp` (visited on 04/13/2022).

[68] YAML Language Development Team. *YAML Ain’t Markup Language (YAML™) version 1.2*. 2021. URL: `https://yaml.org/spec/1.2.2/` (visited on 04/13/2022).

[69] Charles F. F. Karney. “Algorithms for geodesics”. In: *Journal of Geodesy* 87.1 (June 2012), pp. 43–55. DOI: `10.1007/s00190-012-0578-z`. URL: `https://doi.org/10.1007%2Fs00190-012-0578-z`.

[70] Charles Karney. *GeographicLib*. 2022. URL: `https://geographiclib.sourceforge.io/` (visited on 04/15/2022).

[71] W. Fisher and B. Cash. *IEEE 802.11p Draft Review*. Tech. rep. Sept. 2004.

[72] OpenStax College. *Introductory Statistics*. ST-1-000-RS. 2013. ISBN: 9781938168208.

[73] Utah State University. *t-distribution table*. URL: `https://www.usu.edu/math/cfairbourn/Stat2300/t-table.pdf` (visited on 04/17/2022).

# Appendices

# Appendix 1

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

#ifndef RULE_READER_HPP
#define RULE_READER_HPP

#include "signature.hpp"
#include <yaml-cpp/yaml.h>
#include <iostream>


namespace mitvane
{

enum class MetaReaderStatusCode
{
    Success,
    Undefined_Msg,
    Bad_Msg,
    Undefined_Sid,
    Bad_Sid,
    Undefined_Rev,
    Bad_Rev
};

enum class RuleReaderStatusCode
{
```

```cpp
    Success,
    Broken_Rule_Structure,
    Undefined_Action,
    Bad_Action,
    Undefined_Protocol,
    Bad_Protocol,
    Undefined_MetaData,
    Bad_MetaData,
    Bad_Pattern
};

class RuleReader
{
    public:
        RuleReader(std::string rule_filepath);
        RuleReaderStatusCode read(std::map<Protocol, std::vector<Signature>> &signatures
            );
    private:
        Action read_action(YAML::Node action_node);
        Protocol read_protocol(YAML::Node protocol_node);
        MetaReaderStatusCode read_metadata(YAML::Node meta_node, MetaData &meta);

        std::string m_rule_filepath;

};

} // namespace mitvane

#endif // RULE_READER_HPP
```

Listing 1. Rule Reader (header file)

```cpp
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */


#include "rule_reader.hpp"
#include "signature.hpp"
#include "pattern_reader.hpp"
#include <yaml-cpp/yaml.h>
#include <algorithm>
#include <iostream>
```

```


namespace mitvane
{

RuleReader::RuleReader(std::string rule_filepath) :
    m_rule_filepath(rule_filepath){}

Action RuleReader::read_action(YAML::Node action_node)
{
    std::string action = action_node.as<std::string>();
    transform(action.begin(), action.end(), action.begin(), ::tolower);
    if (action == "alert") {
        return Action::Alert;
    } else if (action == "drop") {
        return Action::Drop;
    } else {
        return Action::NotSupported_Action;
    }
}

Protocol RuleReader::read_protocol(YAML::Node protocol_node)
{
    std::string protocol = protocol_node.as<std::string>();
    transform(protocol.begin(), protocol.end(), protocol.begin(), ::tolower);

    if (protocol == "geonetworking" || protocol == "geonet") {
        return Protocol::GeoNetworking;
    } else if (protocol == "btp") {
        return Protocol::BTP;
    } else if (protocol == "facility") {
        return Protocol::Facility;
    } else if (protocol == "ca" || protocol == "cam") {
        return Protocol::CA;
    } else if (protocol == "den" || protocol == "denm") {
        return Protocol::DEN;
    } else if (protocol == "spat" || protocol == "spatem" || protocol == "tlm") {
        return Protocol::SPAT;
    } else if (protocol == "map" || protocol == "mapem" || protocol == "rlt") {
        return Protocol::MAP;
    } else {
        return Protocol::NotSupported_Protocol;
    }
}

MetaReaderStatusCode RuleReader::read_metadata(YAML::Node meta_node, MetaData &meta)
{
    // read msg
    YAML::Node msg_node = meta_node["msg"];

    if (!msg_node.IsDefined()) {
        return MetaReaderStatusCode::Undefined_Msg;
    }
    if (!msg_node.IsScalar()) {
        return MetaReaderStatusCode::Bad_Msg;
    }

    meta.msg = msg_node.as<std::string>();

    // read id
    YAML::Node sid_node = meta_node["sid"];

    if (!sid_node.IsDefined()) {
        return MetaReaderStatusCode::Undefined_Sid;
    }
    if (!sid_node.IsScalar()) {
        return MetaReaderStatusCode::Bad_Sid;
    }

    meta.sid = sid_node.as<uint32_t>();

    // read rev
    YAML::Node rev_node = meta_node["rev"];

    if (!rev_node.IsDefined()) {
        return MetaReaderStatusCode::Undefined_Sid;
    }
    if (!rev_node.IsScalar()) {
        return MetaReaderStatusCode::Bad_Sid;
    }

    meta.rev = rev_node.as<uint16_t>();

    return MetaReaderStatusCode::Success;
}

RuleReaderStatusCode RuleReader::read(std::map<Protocol, std::vector<Signature>> &
    signatures)
{
    YAML::Node ruleset = YAML::LoadFile(m_rule_filepath);
    if (!ruleset.IsMap()) {
        return RuleReaderStatusCode::Broken_Rule_Structure;
    }

    YAML::Node rules = ruleset["rules"];
    if (!rules.IsSequence()) {
        return RuleReaderStatusCode::Broken_Rule_Structure;
    }

    for(int i = 0; i < (int)rules.size(); ++i) {
        Signature sig;

        // read action
        YAML::Node action_node = rules[i]["action"];
        if (!action_node.IsDefined()) {
            return RuleReaderStatusCode::Undefined_Action;
        }
        if (!action_node.IsScalar()) {
            return RuleReaderStatusCode::Bad_Action;
        }

        Action action = read_action(action_node);
        if (action == Action::NotSupported_Action) {
            return RuleReaderStatusCode::Bad_Action;
        }
        sig.action = action;

        // read protocol
        YAML::Node protocol_node = rules[i]["protocol"];
        if (!protocol_node.IsDefined()) {
            return RuleReaderStatusCode::Undefined_Protocol;
        }

        if (!protocol_node.IsScalar()) {
            return RuleReaderStatusCode::Bad_Protocol;
        }

        Protocol protocol = read_protocol(protocol_node);
        if (protocol == Protocol::NotSupported_Protocol) {
            return RuleReaderStatusCode::Bad_Protocol;
        }
        sig.protocol = protocol;

        // read metadata
        YAML::Node meta_node = rules[i]["meta"];
        if (!meta_node.IsDefined()) {
            return RuleReaderStatusCode::Undefined_MetaData;
        }

        if (!meta_node.IsMap()) {
            return RuleReaderStatusCode::Bad_MetaData;
        }

        MetaData meta;
        MetaReaderStatusCode ret = read_metadata(meta_node, meta);
        if (ret != MetaReaderStatusCode::Success) {
            return RuleReaderStatusCode::Bad_MetaData;
        }


        PatternReader pattern_reader = PatternReader();
        PatternReaderStatusCode pattern_st;
        switch (protocol) {
            case Protocol::GeoNetworking:
            {
                pattern_st = pattern_reader.read_geonet_pattern(rules, sig);
                if (pattern_st != PatternReaderStatusCode::Success){
                    return RuleReaderStatusCode::Bad_Pattern;
                }
                if(signatures.count(Protocol::GeoNetworking) == 0) {
                    signatures[Protocol::GeoNetworking] = std::vector<Signature>{sig};
                } else {
                    signatures[Protocol::GeoNetworking].emplace_back(sig);
                }
                break;
            }
            case Protocol::BTP:
            {
                break;
            }
            case Protocol::Facility:
            {
                pattern_st = pattern_reader.read_facility_pattern(rules, sig);
                if (pattern_st != PatternReaderStatusCode::Success){
                    return RuleReaderStatusCode::Bad_Pattern;
                }
                if(signatures.count(Protocol::Facility) == 0) {
                    signatures[Protocol::Facility] = std::vector<Signature>{sig};
                } else {
                    signatures[Protocol::Facility].emplace_back(sig);
                }
                break;
            }
            case Protocol::CA:
            {
                break;
            }
            case Protocol::DEN:
            {
                break;
            }
            case Protocol::MAP:
            {
                break;
            }
            case Protocol::SPAT:
            {
                break;
            }
            default:
                break;
        }
    }
    return RuleReaderStatusCode::Success;
}

} // namespace mitvane
```

Listing 2. Rule Reader (cpp file)

```
/*
 * This file is modified
 * by Tomoya Tanaka <deepsky2221@gmail.com>
 * from <https://github.com/riebl/vanetza/blob/master/vanetza/geonet/router.hpp>
 * at 2022-02-25.
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

/*
 * State Changes
 * - Several functions which are not necessary for decoding are removed.
 * - Data store for detection module in IDS/IPS are added.
 * - Function names are changed from indicate_[basic|common|extended|secured]
 *   to decode_[basic|common|extended|secured]
*/


#ifndef DECODER_HPP
#define DECODER_HPP

#include "mitvane/detector/detection_context.hpp"
#include "mitvane/detector/geonet_data.hpp"
#include "mitvane/decoder/btp_decoder.hpp"



namespace mitvane
{

class GeonetDecoder
{
    typedef std::unique_ptr<vanetza::UpPacket> UpPacketPtr;

    public:
        GeonetDecoder(const vanetza::geonet::MIB&, DetectionContext&);

        void decode(UpPacketPtr, const vanetza::MacAddress& sender, const vanetza::
            MacAddress& destination);

    private:

        void decode_basic(vanetza::geonet::IndicationContextBasic&);
        void decode_common(vanetza::geonet::IndicationContext&, const vanetza::geonet::
            BasicHeader&);
        void decode_extended(vanetza::geonet::IndicationContext&, const vanetza::geonet
            ::CommonHeader&);
        void decode_secured(vanetza::geonet::IndicationContextBasic&, const vanetza::
            geonet::BasicHeader&);

        void pass_up(const vanetza::geonet::DataIndication&, UpPacketPtr);

        const vanetza::geonet::MIB& m_mib;
        DetectionContext& m_detection_context;

};

} // namespace mitvane

#endif //DECODER_HPP
```

Listing 3. Decoder for GeoNetworking packet (hpp file)

```
/*
 * This file is modified
 * by Tomoya Tanaka <deepsky2221@gmail.com>
 * from <https://github.com/riebl/vanetza/blob/master/vanetza/geonet/router.cpp>
 * at 2022-02-25.
 *
 * This file is part of Mitvane.
 *
```

```cpp
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

/*
 * State Changes
 * - Several functions which are not necessary for decoding are removed.
 * - Data storing for detection module in IDS/IPS are added.
 * - Function names are changed from indicate_[basic|common|extended|secured]
 *   to decode_[basic|common|extended|secured]
*/


#include "geonet_decoder.hpp"
#include <vanetza/geonet/indication_context.hpp>



using namespace vanetza;
using namespace vanetza::geonet;
using namespace vanetza::security;

namespace mitvane {

GeonetDecoder::GeonetDecoder(const vanetza::geonet::MIB& mib, DetectionContext&
    detection_context) :
    m_mib(mib), m_detection_context(detection_context){}

void inspect_signer_info(const SecuredMessage* secured_message, DetectionContext&
    detection_context)
{
    detection_context.geonet_data.payload_type = secured_message->payload.type;
    detection_context.geonet_data.secure_protocol_version = secured_message->
        protocol_version();

    const SignerInfo* signer_info = secured_message->header_field<HeaderFieldType::
        Signer_Info>();

    if(signer_info){
        HashedId8 signer_hash;
        signer_hash.fill(0x00);
        SignerInfoType signer_info_type = get_type(*signer_info);
        detection_context.geonet_data.signer_info_type = signer_info_type;
        switch(signer_info_type){
            case SignerInfoType::Certificate:
            {
                signer_hash = calculate_hash(boost::get<Certificate>(*signer_info));
                detection_context.geonet_data.certificate_id = signer_hash;
```

```
                    break;
                }
                case SignerInfoType::Certificate_Digest_With_SHA256:
                {
                    signer_hash = boost::get<HashedId8>(*signer_info);
                    detection_context.geonet_data.certificate_id = signer_hash;
                    break;
                }
                case SignerInfoType::Certificate_Chain:
                {
                    std::list<Certificate> chain = boost::get<std::list<Certificate>>(*
                        signer_info);
                    detection_context.geonet_data.chain_size = chain.size();
                    signer_hash = calculate_hash(chain.back());
                    detection_context.geonet_data.certificate_id = signer_hash;
                    break;
                }
                default:
                    // logging
                    break;
            }
        }

}

void GeonetDecoder::decode(UpPacketPtr packet, const vanetza::MacAddress& sender, const
    vanetza::MacAddress& destination)
{
    assert(packet);

    struct indication_visitor : public boost::static_visitor<>
    {
        indication_visitor(GeonetDecoder& decoder, const IndicationContext::LinkLayer&
            link_layer, UpPacketPtr packet) :
            m_decoder(decoder), m_link_layer(link_layer), m_packet(std::move(packet))
        {
        }

        void operator()(vanetza::CohesivePacket& packet)
        {
            IndicationContextDeserialize ctx(std::move(m_packet), packet, m_link_layer);
            m_decoder.decode_basic(ctx);
        }

        void operator()(vanetza::ChunkPacket& packet)
        {
            IndicationContextCast ctx(std::move(m_packet), packet, m_link_layer);
            m_decoder.decode_basic(ctx);
        }

        GeonetDecoder& m_decoder;
        const IndicationContext::LinkLayer& m_link_layer;
        UpPacketPtr m_packet;
    };

    IndicationContext::LinkLayer link_layer;
    link_layer.sender = sender;
    link_layer.destination = destination;

    UpPacket* packet_ptr = packet.get();
    indication_visitor visitor(*this, link_layer, std::move(packet));
    boost::apply_visitor(visitor, *packet_ptr);
}

void GeonetDecoder::decode_basic(IndicationContextBasic& ctx)
{
    const BasicHeader* basic = ctx.parse_basic();
    if (!basic) {
        // logging
    } else {
        m_detection_context.geonet_data.protocol_version = basic->version.raw();
        // Store lifetime
        m_detection_context.geonet_data.remaining_packet_lifetime = basic->lifetime;
        // Store RHL (Remaining Hop Limit)
        m_detection_context.geonet_data.remaining_hop_limit = basic->hop_limit;

        m_detection_context.geonet_data.next_header = basic->next_header;

        if (basic->next_header == NextHeaderBasic::Secured) {
            m_detection_context.geonet_data.security_info = SecurityInfo::Signed;
            decode_secured(ctx, *basic);
        } else if (basic->next_header == NextHeaderBasic::Common) {
            m_detection_context.geonet_data.security_info = SecurityInfo::NoSecurity;
            decode_common(ctx, *basic);
        } else {
            // logging
        }
    }
}

void GeonetDecoder::decode_common(IndicationContext& ctx, const BasicHeader& basic)
{
    const CommonHeader* common = ctx.parse_common();
    if (!common) {
        // logging
    } else {
        DataIndication& indication = ctx.service_primitive();
        indication.traffic_class = common->traffic_class;
        m_detection_context.geonet_data.header_type = common->header_type;
        m_detection_context.geonet_data.maximum_hop_limit = common->maximum_hop_limit;
        switch (common->next_header)
        {
            case NextHeaderCommon::BTP_A:
                indication.upper_protocol = UpperProtocol::BTP_A;
                break;
            case NextHeaderCommon::BTP_B:
                indication.upper_protocol = UpperProtocol::BTP_B;
                break;
            case NextHeaderCommon::IPv6:
                indication.upper_protocol = UpperProtocol::IPv6;
                break;
            default:
                indication.upper_protocol = UpperProtocol::Unknown;
                break;
        }

        // execute steps depending on extended header type
        decode_extended(ctx, *common);
```

```cpp
    }
}

void GeonetDecoder::decode_secured(IndicationContextBasic& ctx, const BasicHeader& basic
    )
{

    struct secured_payload_visitor : public boost::static_visitor<>
    {
        secured_payload_visitor(GeonetDecoder& decoder, IndicationContextBasic& ctx,
            const BasicHeader& basic) :
            m_decoder(decoder), m_context(ctx), m_basic(basic)
        {
        }

        void operator()(ChunkPacket& packet)
        {
            IndicationContextSecuredCast ctx(m_context, packet);
            m_decoder.decode_common(ctx, m_basic);
        }

        void operator()(CohesivePacket& packet)
        {
            IndicationContextSecuredDeserialize ctx(m_context, packet);
            m_decoder.decode_common(ctx, m_basic);
        }

        GeonetDecoder& m_decoder;
        IndicationContextBasic& m_context;
        const BasicHeader& m_basic;
    };

    auto secured_message = ctx.parse_secured();
    if (!secured_message) {
        // logging
    } else {
        inspect_signer_info(secured_message, m_detection_context);
        secured_payload_visitor visitor(*this, ctx, basic);
        PacketVariant plaintext_payload = std::move(secured_message->payload.data);
        boost::apply_visitor(visitor, plaintext_payload);
    }
}

void GeonetDecoder::decode_extended(IndicationContext& ctx, const CommonHeader& common)
{
    struct extended_header_visitor : public boost::static_visitor<>
    {
        extended_header_visitor(GeonetDecoder& router, IndicationContext& ctx, const
            UpPacket& packet, mitvane::DetectionContext& detection_context) :
            m_decoder(router), m_context(ctx), m_packet(packet), m_detection_context(
                detection_context)
        {
        }

        void operator()(const ShbHeader& shb)
        {
            DataIndication& indication = m_context.service_primitive();
            indication.transport_type = TransportType::SHB;
            indication.source_position = static_cast<ShortPositionVector>(shb.
                source_position);

            // Store Source Position Vector
            m_detection_context.geonet_data.source_position = static_cast<
                ShortPositionVector>(shb.source_position);
        }

        void operator()(const GeoBroadcastHeader& gbc)
        {
            DataIndication& indication = m_context.service_primitive();
            indication.transport_type = TransportType::GBC;
            indication.source_position = static_cast<ShortPositionVector>(gbc.
                source_position);
            indication.destination = gbc.destination(m_context.pdu().common().
                header_type);

            // Store Source Position Vector
            m_detection_context.geonet_data.source_position = static_cast<
                ShortPositionVector>(gbc.source_position);
            // Store destination
            m_detection_context.geonet_data.destination = gbc.destination(m_context.pdu
                ().common().header_type);
        }

        void operator()(const BeaconHeader& beacon)
        {
        }

        GeonetDecoder& m_decoder;
        IndicationContext& m_context;
        const UpPacket& m_packet;
        mitvane::DetectionContext& m_detection_context;
    };

    auto extended = ctx.parse_extended(common.header_type);
    UpPacketPtr packet = ctx.finish();
    // Store payload data
    m_detection_context.payload = packet.get();
    assert(packet);

    if (!extended) {
        // logging
    } else {
        extended_header_visitor visitor(*this, ctx, *packet, m_detection_context);
        boost::apply_visitor(visitor, *extended);
        pass_up(ctx.service_primitive(), std::move(packet));
    }
}

void GeonetDecoder::pass_up(const DataIndication& ind, UpPacketPtr packet)
{
    BtpDecoder btp_decoder(m_detection_context);
    btp_decoder.decode(ind, std::move(packet));
}

} // namespace mitvane
```

Listing 4. Decoder for GeoNetworking packet (cpp file)

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

#ifndef CLASSIFIER_HPP
#define CLASSIFIER_HPP

#include "mitvane/detector/detection_context.hpp"
#include "mitvane/app_layer_parser/application_parser.hpp"
#include <vanetza/btp/header.hpp>
#include <unordered_map>


namespace mitvane {


class Classifier {
    public:
        typedef std::unordered_map<vanetza::btp::port_type, ApplicationLayerParser*>
            port_map;

        Classifier(DetectionContext&);
        ApplicationLayerParser* classify(port_map&);

    private:
        DetectionContext& m_detection_context;
};

} // namespace mitvane

#endif // CLASSIFIER_HPP
```

Listing 5. Classifier (hpp file)

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */


#include "classifier.hpp"
#include <cassert>


using namespace vanetza;
using namespace vanetza::btp;


namespace mitvane
{

Classifier::Classifier(DetectionContext& detection_context) :
    m_detection_context(detection_context){}

ApplicationLayerParser* Classifier::classify(port_map& app_layer_parsers)
{
    ApplicationLayerParser* handler = nullptr;

    handler = app_layer_parsers[m_detection_context.app_layer_parser_port];

    return handler;

}
} // namespace mitvane
```

Listing 6. Classifier (cpp file)

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

```

```

#ifndef GEONET_DETECTOR_HPP
#define GEONET_DETECTOR_HPP

#include "mitvane/idps_context.hpp"
#include "detector.hpp"
#include "geonet_data.hpp"

namespace mitvane
{

enum class TransformStatusCode {
    Success,
    Error
};

class GeonetDetector : public Detector
{

    public:
        GeonetDetector (GeonetData& data, vanetza::PositionProvider& positioning);
        void detect(signatures_type &signatures) override;

    private:
        TransformStatusCode transform_nh(std::string nh_str, vanetza::geonet::
            NextHeaderBasic &nh);
        TransformStatusCode transform_ht(std::string ht_str, uint8_t &ht);
        GeonetData& m_data;
        vanetza::PositionProvider& m_positioning;
};



} // namespace mitvane

#endif /* GEONET_DETECTOR_HPP */
```

Listing 7. Detector for GeoNetworking protocol (hpp file)

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

```

```cpp

#include "mitvane/idps_context.hpp"
#include "mitvane/rule_reader/rule_reader.hpp"
#include "detector.hpp"
#include "geonet_data.hpp"
#include "geonet_detector.hpp"
#include <vanetza/units/angle.hpp>
#include <vanetza/units/length.hpp>
#include <vanetza/security/region.hpp>
#include <boost/units/cmath.hpp>
#include <GeographicLib/Geodesic.hpp>

namespace mitvane
{

using namespace vanetza::geonet;

GeonetDetector::GeonetDetector(GeonetData &data, vanetza::PositionProvider& positioning)
    :
    m_data(data), m_positioning(positioning){}

bool is_within_circle(const vanetza::geonet::Area& inner, const vanetza::geonet::Area&
    outer)
{
    const auto& geod = GeographicLib::Geodesic::WGS84();
    double center_dist = 0.0;
    const vanetza::units::GeoAngle inner_lat = inner.position.latitude;
    const vanetza::units::GeoAngle inner_long = inner.position.longitude;
    const vanetza::units::GeoAngle outer_lat = outer.position.latitude;
    const vanetza::units::GeoAngle outer_long = outer.position.longitude;

    Circle range = boost::get<Circle>(inner.shape);
    Circle dest_area = boost::get<Circle>(outer.shape);
    geod.Inverse(inner_lat / vanetza::units::degree, inner_long / vanetza::units::degree
        ,
            outer_lat / vanetza::units::degree, outer_long / vanetza::units::degree,
                center_dist);
    return center_dist + range.r / vanetza::units::si::meter <= dest_area.r / vanetza::
        units::si::meter;
}

TransformStatusCode GeonetDetector::transform_nh(std::string nh_str, NextHeaderBasic &nh
    )
{
    if (nh_str == "common") {
        nh = NextHeaderBasic::Common;
        return TransformStatusCode::Success;
    } else if (nh_str == "secure") {
        nh = NextHeaderBasic::Secured;
        return TransformStatusCode::Success;
    } else {
        return TransformStatusCode::Error;
    }
}

TransformStatusCode GeonetDetector::transform_ht(std::string ht_str, uint8_t &ht){
    if (ht_str == "TSB") {
        ht = static_cast<uint8_t>(HeaderType::TSB_Multi_Hop);
        return TransformStatusCode::Success;
    } else if (ht_str == "SHB") {
        ht = static_cast<uint8_t>(HeaderType::TSB_Single_Hop);
        return TransformStatusCode::Success;
    } else if (ht_str == "GBC") {
        ht = static_cast<uint8_t>(HeaderType::GeoAnycast_Circle) |
            static_cast<uint8_t>(HeaderType::GeoAnycast_Rect) |
            static_cast<uint8_t>(HeaderType::GeoAnycast_Elip);
    } else {
        return TransformStatusCode::Error;
    }
}

void GeonetDetector::detect(signatures_type &signatures)
{
    std::vector<mitvane::Signature> geonet_signatures = signatures[Protocol::
        GeoNetworking];
    TransformStatusCode ret;

    for (auto sig_it = geonet_signatures.begin(); sig_it != geonet_signatures.end(); ++
        sig_it) {
        mitvane::Signature sig = *sig_it;

        if (sig.patterns.count("nh")) {
            NextHeaderBasic nh;
            ret = transform_nh(boost::get<std::string>(sig.patterns["nh"]), nh);
            if (ret != TransformStatusCode::Success) {
                std::cerr << "[sid: " << sig.meta.sid << "] " << "Detection skipped:
                    Invalid next header" << "\n";
                continue;
            }
            if (nh != m_data.next_header) {
                //message_handler.handle(action, msg);
            }
        }
        if (sig.patterns.count("ht")){
            uint8_t ht;
            ret = transform_ht(boost::get<std::string>(sig.patterns["ht"]), ht);
            if (ret != TransformStatusCode::Success) {
                std::cerr << "[sid: " << sig.meta.sid << "] " << "Detection skipped:
                    Invalid header type" << "\n";
                continue;
            }
            if (!(ht & static_cast<uint8_t>(m_data.header_type))) {
                //message_handler.handle(action, msg);
            }
        }
        if (sig.patterns.count("mhl")){
            if(boost::get<int>(sig.patterns["mhl"]) <= m_data.maximum_hop_limit) {
                //message_handler.handle(action, msg);
            }
        }
        if (sig.patterns.count("destination_area") && m_data.destination) {
            DestinationVariant dest = *m_data.destination;
            vanetza::units::Length limit_a
                = boost::get<geonet_destination_area>(sig.patterns["destination_area"]).
                    distance_a * boost::units::si::meters;
            vanetza::units::Length limit_b
                = boost::get<geonet_destination_area>(sig.patterns["destination_area"]).
                    distance_b * boost::units::si::meters;

            if(dest.type() == typeid(Area)){
                Area dest_area = boost::get<Area>(dest);
                if (dest_area.shape.type() == typeid(Circle)) {
                    Circle dest_area_circle = boost::get<Circle>(dest_area.shape);
                    if (limit_a < dest_area_circle.r){
                        //message_handler.handle(action, msg)
                    }
                } else if (dest_area.shape.type() == typeid(Rectangle)) {
                    Rectangle dest_area_rect = boost::get<Rectangle>(dest_area.shape);
                    if (limit_a < dest_area_rect.a || limit_b < dest_area_rect.b){
                        //message_handler.handle(sig)
                    }
                } else if (dest_area.shape.type() == typeid(Ellipse)) {
                    Ellipse dest_area_ellipse = boost::get<Ellipse>(dest_area.shape);
                    if (limit_a < dest_area_ellipse.a || limit_b < dest_area_ellipse.b){
                        //message_handler.handle(action, msg)
                    }
                } else {
                    std::cout << "[sid: " << sig.meta.sid << "]" << "Detection skipped:
                        Invalid destination shape" << "\n";
                    continue;
                }

            } else {
                std::cout << "[sid: " << sig.meta.sid << "]" << "Detection Skipped: area
                     is not included in the destination field" << "\n";
                continue;
            }
        }
        if (sig.patterns.count("allowed_so_range")) {
            geonet_allowed_so_range allowed_so_range = boost::get<
                geonet_allowed_so_range>(sig.patterns["allowed_so_range"]);
            vanetza::units::Length limit_a
                = boost::get<geonet_destination_area>(sig.patterns["allowed_so_range"]).
                    distance_a * boost::units::si::meters;
            vanetza::units::Length limit_b
                = boost::get<geonet_destination_area>(sig.patterns["allowed_so_range"]).
                    distance_b * boost::units::si::meters;
            if (allowed_so_range.shape == Allowed_So_Range_Shape::Circle) {
                vanetza::security::TwoDLocation range_center
                    = vanetza::security::TwoDLocation(m_positioning.position_fix().
                        latitude, m_positioning.position_fix().longitude);
                vanetza::security::CircularRegion allowed_range_circle = vanetza::
                    security::CircularRegion(range_center, limit_a);

                vanetza::geonet::geo_angle_i32t source_latitude = m_data.source_position
                    .latitude;
                vanetza::geonet::geo_angle_i32t source_longitude = m_data.
                    source_position.longitude;
                vanetza::security::TwoDLocation source_position
                    = vanetza::security::TwoDLocation(source_latitude, source_longitude)
                        ;

                if (!vanetza::security::is_within(source_position, allowed_range_circle)
                    ) {
                     //message_handler.handle(action, msg)
                }

            } else {
                std::cout << "[sid: " << sig.meta.sid << "]" << "Detection Skipped: " <<
                    "Shapes except for circle as allowed_so_range's shape is currently not supported" << "\n";
                continue;
            }


        }

        //"nh", "ht", "mhl", "destination_area", "allowed_so_range"


    }

}



} // namespace mitvane
```

Listing 8. Detector for GeoNetworking protocol (cpp file)

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

#ifndef MESSAGE_HANDLER_HPP
#define MESSAGE_HANDLER_HPP

#include "logging.hpp"
#include "mitvane/rule_reader/signature.hpp"

namespace mitvane
{

enum class HandleReport {
    Allow,
    Drop
};

class MessageHandler
{
    public:
        MessageHandler() = default;
        HandleReport handle(std::vector<Signature>& sigs);
    private:
        boost::log::sources::logger lg;
};

} // namespace mitvane

#endif //MESSAGE_HANDLER_HPP
```

Listing 9. Message Handler (hpp file)

```
/*
 * (C) 2022 Tomoya Tanaka <deepsky2221@gmail.com>
 *
 * This file is part of Mitvane.
 *
 * Mitvane is free software: you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published
 * by the Free Software Foundation, either version 3 of the License, or
 * any later version.
 * Mitvane is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License and
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * and GNU Lesser General Public License along with Mitvane.
 * If not, see <https://www.gnu.org/licenses/>.
 */

#include "logging.hpp"
#include "message_handler.hpp"
#include "mitvane/rule_reader/signature.hpp"

namespace mitvane
{


std::string action_to_string(Action& action){
    if (action == Action::Alert) {
        return "alert";
    } else if (action == Action::Drop) {
        return "drop";
    } else {
        return "unknown action";
    }
}

std::string protocol_to_string(Protocol& protocol){
    if (protocol == Protocol::GeoNetworking) {
        return "GeoNetworking";
    } else if (protocol == Protocol::BTP) {
        return "BTP";
    } else if (protocol == Protocol::Facility) {
        return "Facility";
    } else if (protocol == Protocol::CA) {
```

```cpp
        return "CA";
    } else if (protocol == Protocol::DEN) {
        return "DEN";
    } else if (protocol == Protocol::SPAT) {
        return "SPAT";
    } else if (protocol == Protocol::MAP) {
        return "MAP";
    } else {
        return "unknown protocol";
    }
}

std::string sid_rev_to_string(MetaData& meta) {
    return "[" + std::to_string(meta.sid) + ":" + std::to_string(meta.rev) + "] ";
}


std::string format_log(Signature& signature) {
    return sid_rev_to_string(signature.meta) +
        signature.meta.msg +
        + " " + action_to_string(signature.action) + " " +
        "{" + protocol_to_string(signature.protocol) + "}";
}

HandleReport MessageHandler::handle(std::vector<Signature>& sigs) {
    HandleReport rep = HandleReport::Allow;
    for (auto sig = sigs.begin(); sig != sigs.end(); ++sig) {
        if (sig->action == Action::Alert) {
            BOOST_LOG(lg) << format_log(*sig);
        } else if (sig->action == Action::Drop) {
            BOOST_LOG(lg) << format_log(*sig);
            rep = HandleReport::Drop;
        } else {
            BOOST_LOG(lg) << "Unknown action. No log output.";
            rep = HandleReport::Drop;
        }
    }
    return rep;
}

} // namespace mitvane
```

Listing 10. Message Handler (cpp file)