Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

SUCCESS-6G: DEVISE

WP3 Deliverable E8

Enablers for intelligent security enforcement in V2X systems

Project Title:	SUCCESS-6G: DEVISE

Title of Deliverable:	Enablers for intelligent security enforcement in V2X systems
Status-Version:	Final-1.0
Delivery Date:	31/03/2025

Contributors:	Roshan Sedar, Charalampos Kalalas, Pol Alemany, Ricard Vilalta, Raul Muñoz (CTTC), Guillermo Candela Belmonte, Antonio Fernandez (Optare)
Lead editor::	Charalampos Kalalas (CTTC)
Reviewers:	Miquel Payaro, Charalampos Kalalas (CTTC)
Keywords:	V2X security, Trustworthiness, Zero-touch network and service management, Transfer learning, Deep reinforcement learning, Misbehavior detection

Document Revision History

Version	Date	Description
v0.1	10 February 2025	Initial content addded
v0.2	21 February 2025	Main content addded
v0.3	10 March 2025	Revision and additional content added
v1.0	31 March 2025	Approved form of Deliverable E8

Acknowledgement
The research conducted by SUCCESS-6G - TSI-063000-2021-39/40/41 receives funding from the Ministerio de Asuntos Económicos y Transformación Digital and the European Union-NextGenerationEU under the framework of the "Plan de Recuperación, Transformación y Resiliencia" and the "Mecanismo de Recuperación y Resiliencia".

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive summary

This deliverable introduces the final set of security enablers specifically designed to support intelligent, context-aware security enforcement in Vehicle-to-Everything (V2X) systems. These enablers form a foundational layer for enhancing trust and resilience, encompassing attack detection methods and trust-aware knowledge exchange in decentralized V2X environments. By integrating intelligence and context awareness into the security decision-making process, SUCCESS-6G-DEVISE facilitates security enforcement by enhancing responsiveness to emerging threats.

Table of Contents

1 Introduction

This deliverable presents a comprehensive suite of security enablers specifically engineered to support intelligent and context-aware security enforcement within Vehicle-to-Everything (V2X) communication systems. These enablers constitute a critical foundation for enhancing trust, resilience, and situational awareness in vehicular environments, where rapid decision-making and dynamic network topologies present unique security challenges. The proposed mechanisms include advanced misbehavior detection techniques, as well as trust-aware knowledge exchange schemes that facilitate secure information sharing among edge nodes in a distributed setting.

A key innovation of the SUCCESS-6G-DEVISE project lies in the integration of contextual intelligence into the security decision-making process. As such, V2X security posture can be dynamically adjusted—allowing for more adaptive, fine-grained attack detection strategies. This results in enhanced responsiveness to both known and previously unseen security threats, improving the overall robustness of V2X systems against a wide range of attacks. Through these capabilities, the proposed enablers significantly contribute to the realization of security enforcement in 6G-enabled V2X ecosystems, aligning with the project's broader objective of securing V2X communication for next-generation mobility.

1.1 Structure of the document

Section 2 presents a collaborative misbehavior detection methodology leveraging deep reinforcement learning for misbehavior detection and building on trust-aware knowledge exchange among geographically distributed roadside units (RSUs). Section 3 demonstrates the feasibility of a zero-touch network and service management (ZSM)–based framework to autonomously protect V2X services at the edge by effectively mitigating various distributed denial-of-service (DDoS) attacks. Finally, Section 4 introduces a hybrid deep learning methodology for misbehavior detection, where *i*) unsupervised learning is used to adapt to the dynamic nature of V2X traffic patterns, overcoming the scarcity of labeled data for attacks, and *ii*) supervised learning is employed to refine the classification and achieve high accuracy in real time.

2 Deep Reinforcement Learning for Untrusted Distributed Environments in Automotive Use Cases

2.1 Introduction

The recent advancements in V2X technology promise increased road safety, driving autonomy, and inclusive mobility options. Inevitably, this evolution has given rise to new threat vectors associated with the inherent V2X security vulnerabilities, which adversaries may maliciously exploit to disrupt system operation [1]. Among them, misbehavior attacks launched by rogue insiders often become difficult to detect and contain, since malicious nodes may alter their activity intelligently over time [2]. Although cryptographic techniques are capable of limiting outsiders by offering authentication, integrity, and non-repudiation as a first layer of defense, they may fall short in detecting rogue or dishonest behavior and identifying V2X insiders with malicious intent. As such, the trustworthiness of exchanged information cannot be guaranteed.

Emerging data-driven approaches driven by artificial intelligence and machine learning (AI/ML) tools provide a fertile ground for addressing misbehavior attacks [3]. With the availability of vehicular data streams, AI/ML-based schemes can facilitate the analysis of behavioral patterns for V2X entities and determine trustworthiness levels. Such capabilities have the potential to overcome the shortcomings of traditional misbehavior countermeasures, offering effective solutions to achieve demanding security requirements. Despite the envisioned benefits of data-driven misbehavior detection, the vulnerabilities of AI/ML models introduce additional threat vectors, giving rise to finely targeted, stealthy, and scalable adversarial attacks. Such sophisticated attack types may target both model training (i.e., poisoning attacks) and test (i.e., evasion attacks) phases [4], and undermine the efficacy of misbehavior detection. Collaborative misbehavior detection, relying on decentralized learning models at the vehicular edge, may further extend the attack surface and, thus, exacerbate the impact of adversarial attacks. Consequently, the pervasive adoption of AI/ML models for misbehavior detection could be hindered if security concerns related to the model vulnerabilities are not addressed.

Considering the aforementioned research challenges, we introduce a novel approach for collaborative misbehavior detection that builds on geographically distributed RSUs. Each RSU employs a deep reinforcement learning (DRL) model for the detection of malicious traffic stemming from misbehaving vehicles. Leveraging transfer learning principles, the knowledge learned at source RSUs is shared with the target RSU to reuse relevant expertise for misbehavior detection. In the presence of adversarial attacks, the proposed approach performs selective knowledge transfer from trustworthy source RSUs to avoid negative knowledge sharing from adversary-influenced RSUs. For this purpose, a novel trust evaluation metric, referred to as semantic relatedness, is used by the target RSU to quantify the trust level of each source RSU for collaborative misbehavior detection. Through diverse scenarios of collaborative misbehavior detection and an open-source dataset, we evaluate the learning performance of involved RSUs in the presence of adversaries. Besides reducing the training time at the target RSU, our scheme is shown to significantly outperform the baseline scheme with tabula rasa learning, demonstrating its high effectiveness. Interestingly, our approach enhances robustness and generalizability by effectively detecting previously unseen and partially observable misbehaviors.

2.1.1 Related Work

Aiming to address the complex V2X security landscape, several recent works leverage AI/ML-driven techniques for misbehavior detection [3]. Supervised learning schemes may be impractical in V2X scenarios with an expanded attack surface, due to limited access to labeled training examples and/or dependence on security threshold values. On the other hand, unforeseen alterations in vehicular mobility, due to either naturally drifting traffic patterns or unprecedented malicious activity, introduce challenges (e.g., model overfitting) to misbehavior detection schemes relying on conventional deep learning (DL). Furthermore, a number of reputation or trust-based approaches have been proposed in relevant literature to elevate trustworthiness levels in untrusted vehicular environments [5]. Yet, their applicability in collaborative misbehavior detection may be limited, since such methods often assume infrastructure nodes (e.g., RSUs) to be legitimate and non-susceptible to adversarial attacks.

It is plausible that decentralized learning architectures for collaborative misbehavior detection inadvertently make the underlying models attractive targets for adversarial attacks. In turn, locally trained misbehavior detection models can be influenced to reach incorrect decisions/predictions or leak confidential information. For instance, in data poisoning attacks [6], an adversary aims at deliberately modifying the learning algorithm during training via false data injection or manipulation. Surprisingly, the detrimental impact of such adversarial manipulations on misbehavior detection performance remains rather unexplored. Dealing with adversarial attacks is often non-trivial and requires enhanced solutions to foster trust and stimulate confidence in data-driven misbehavior detection.

Overall, existing AI/ML-based misbehavior detection models can be either centrally trained in the cloud and tested locally, or both training and detection are realized in vehicles or RSUs. Yet, these models are susceptible to adversarial attacks, e.g., data poisoning [7]. As such, an attacker may poison the centralized model training pipeline and influence downstream tasks at vehicles or RSUs to misclassify, which could inflict a single point of failure. Similarly, locally trained misbehavior detection models may suffer from data poisoning and adversarial manipulations owing to the extended attack surface. Hence, albeit offering enhanced detection performance, such models may not be robust enough against adversarial attacks. Moreover, they may have limited capability in detecting unseen (i.e., non-anticipated) misbehavior attacks. On the contrary, we propose distributed collaborative learning to enhance misbehavior detection performance, making it robust against adversarial attacks and capable of generalizing to detect unseen and partially observable attacks.

2.2 System Model

This section introduces the vehicular network model, misbehavior detection model, and adversarial model considered in this work. We provide the details in the following.

2.2.1 Vehicular Network Model

Vehicular networks typically comprise a large number of geographically distributed RSUs. RSUs are stationary entities interconnected with each other and the Internet. The usability of RSUs is multifaceted as they offer various services such as Internet access, security solutions, and real-time traffic data distribution. Normally, RSUs have superior computational capabilities

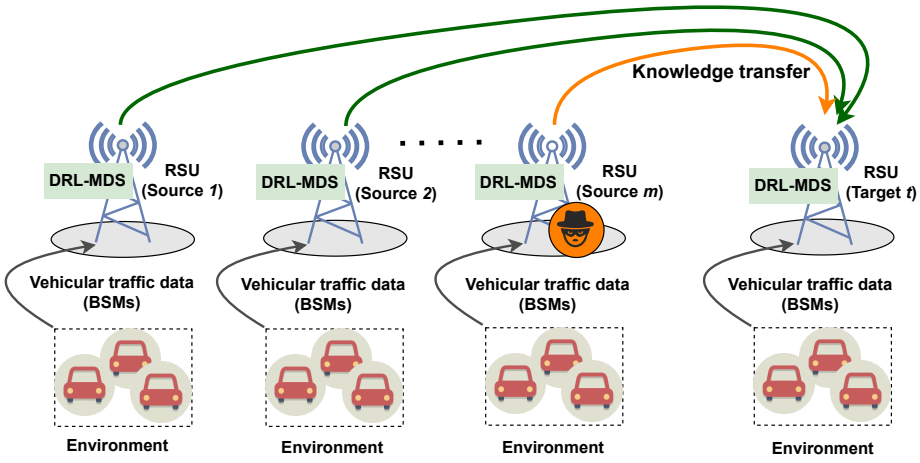

Figure 1: Considered network model for collaborative misbehavior detection. Distributed knowledge transfer between source RSUs $\{s_i\}_{i=1}^{m}$ and the target RSU t is depicted with green arrows for positive knowledge and with orange for negative knowledge [8].

compared to in-vehicle resources, which allows vehicles to offload computation-intensive tasks to RSUs. Figure 1 illustrates the distributed vehicular network model considered in this work. Each RSU experiences its own vehicular environment and receives traffic within its coverage. The RSUs are interconnected via wired connections to provide reliable RSU-to-RSU communication. We assume authentication and authorization procedures have already been performed before V2X communication takes place among entities. The involved vehicles (i.e., authenticated and authorized) periodically broadcast BSMs, which are received by RSUs in their connectivity range. BSMs include standard-related parameters such as position, speed, acceleration, heading angle, and other relevant vehicular information [9].

In our proposed setup, a distributed collaborative misbehavior detection system is considered where we leverage knowledge transfer in the context of transfer learning. As shown in Figure 1, each RSU is equipped with a DRL-based misbehavior detection system (DRL-MDS) and detects misbehaving vehicles. Misbehavior detection is performed at the RSU level, since the vehicle may not have complete information in its communication range due to ephemeral connectivity and/or limited computational resources. The knowledge learned at source RSUs is transferred to the target RSU to reuse existing knowledge during its learning process. The target RSU may not necessarily need to be a neighbor, but it could also be a distant RSU, which shall be reusing the available knowledge of sources to detect misbehaviors.

2.2.2 Misbehavior Detection Model

In our approach, each RSU is equipped with an MDS as shown in Figure 1. The DRL-based model, introduced in our previous works [10, 11], is used for detecting misbehaving vehicles. Here, we present the formulation of the tabula rasa DRL model used for misbehavior detection. A tabula rasa model aims to learn efficiently from scratch without any external or previous knowledge. The details of transfer learning incorporation into the DRL model are provided in Section 2.3.

DRL Model

The vehicular environment considered in this study follows a Markov decision process (MDP) framework to facilitate the detection of misbehaviors through sequential decision-making. Consistent with the MDP formulation, the action of misbehavior detection changes the environment based on the decision of either genuine or malicious behavior at time-step t. Subsequently, the decision at time-step $t+1$ is influenced by the altered environment from the previous time-step t. The aggregated vehicular traffic at each RSU consists of a time-series repository of received BSMs with intrinsic spatiotemporal interdependencies. In this work, we consider a DRL-based misbehavior detector deployed at the edge RSU. The detector (agent) interacts with the vehicular environment to learn the optimal detection policy π^*. During training, the ϵ-greedy method is leveraged to strike a balance between exploration and exploitation in the agent's strategy. Next, we describe the components relevant to the DRL model.

i) Agent: The agent receives the vehicular traffic data as a time series and prior related decisions as inputs (i.e., state $\mathbf{s_t}$), and generates the new decision made (i.e., action a_t) as output. The agent's deep neural network comprises an LSTM layer and a fully connected neural network with linear activation to generate Q-values as choices for the action a_t. At each time-step t, the agent's actions are selected by the policy π. The agent's experience, i.e., $e_t =< \mathbf{s_t}, a_t, r_t, \mathbf{s_{t+1}} >$, stores all the behaviors of the misbehavior detector. By exploiting experience, the detector is progressively improved to obtain a better estimation of the $Q(\mathbf{s}, a)$ function. The objective is to maximize the expected sum of future discounted rewards by learning the π^*. The discounted reward return is expressed as

$$R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k. \tag{1}$$

Q-learning model updates are performed with learning rate α and discount factor γ as

$$\begin{aligned} Q(\mathbf{s_t}, a_t) \leftarrow\ & Q(\mathbf{s_t}, a_t) \\ & + \alpha(r_t + \gamma \max_{a_{t+1}} Q(\mathbf{s_{t+1}}, a_{t+1}) - Q(\mathbf{s_t}, a_t)). \end{aligned} \tag{2}$$

ii) States: The state space comprises the sequence of previous actions denoted by $s_{action} =< a_{t-1}, a_t, ..., a_{t+n-1} >$, and the current BSM information denoted by $s_{time} =< X_t, X_{t+1}, ..., X_{t+n} >$. $\mathbf{X_t} \in \mathbb{R}^d$ is a d-dimensional feature vector at time-step t, including information on d different features. According to the state design, the next action taken by the agent depends on the previous actions and the current BSM information. This design enables the agent to capture temporal dependencies and make more informed decisions.

iii) Actions: The action space is defined as $\mathcal{A} = \{0,1\}$, where 1 indicates the detection of a misbehavior and 0 represents the genuine behavior. The deterministic detection policy π can be expressed as a mapping, i.e., $\pi : \mathbf{s_t} \in \mathcal{S} \longmapsto a_t \in \mathcal{A}$, from states to actions, where $\pi(s)$ prescribes the action that the agent takes at state $\mathbf{s}$. In a given state $\mathbf{s_t}$, the agent selects the action based on the optimal detection policy given by

$$\pi^* = \arg\max_{a \in \mathcal{A}} Q^*(\mathbf{s}, a). \tag{3}$$

iv) Rewards: The reward function $R(\mathbf{s}, a)$ is defined based on the confusion matrix typically used in ML classification problems. A numerical value for r_t is assigned based on the ground

truth information of BSMs. A positive reward is given upon correct detection of a misbehavior, i.e., true positive (TP), or a normal state, i.e., true negative (TN). A negative reward is given when a normal state is incorrectly identified as a misbehavior, i.e., false positive (FP), or a misbehavior as a normal state, i.e., false negative (FN). The agent is penalized more for FN actions than for FPs, as the correct identification of misbehavior is necessary to avoid hazardous situations. Accordingly, we define the immediate reward $r_t \in R$ of the agent as

$$r(\mathbf{s_t}, a_t) = \begin{cases} a, & \text{if } a_t \text{ is a TP,} \\ b, & \text{if } a_t \text{ is a TN,} \\ -c, & \text{if } a_t \text{ is an FP,} \\ -d, & \text{if } a_t \text{ is an FN,} \end{cases} \tag{4}$$

where $a, b, c, d > 0$, with $a > b$ and $d > c$.

2.2.3 Adversarial Model

In this work, we assume the presence of adversaries attempting to contaminate the training data and realize poisoning attacks on the distributed and collaborative DRL-MDS models. In particular, we consider two data poisoning attacks pertinent to adversarial ML: $i)$ label-flipping and $ii)$ policy induction attacks. In the case of label-flipping, it is assumed that the adversaries are rogue insiders who contribute to the training data or have access to the training data itself. In policy induction, we consider an exogenous attacker who can modify the state space before it is observed by the DRL agent. In both cases, attackers aim to maliciously influence the learning model and, subsequently, force incorrect outcomes in downstream misbehavior detection tasks.

In a label-flipping attack, we consider an attacker who targets the malicious class to flip the labels of certain source training data instances at an RSU. In this scenario, the adversarial attacker randomly selects a set of misbehaving vehicles and flips the labels of their data instances into genuine ones, resulting in a targeted random label-flipping attack (Definition 1). The attacker aims to misclassify selected training samples from malicious class 1 to genuine class 0. Following Definition 1, we denote D_{S_i} as the training data of source RSU S_i, i.e., $D_{S_i} = \{(x_{S_{i1}}, y_{S_{i1}}), ..., (x_{S_{ik}}, y_{S_{ik}}), ..., (x_{S_{in}}, y_{S_{in}})\}$, with $x_{S_{ik}} \in X_{S_i}$ representing the k-th data instance of D_{S_i} while $y_{S_{ik}} \in Y_{S_i}$ represents the corresponding label of $x_{S_{ik}}$.

Definition 1 (Label-flipping attack) *Given training samples $\{x_{S_{ik}}, y_{S_{ik}}\}_{k=1}^{n}$ of D_{S_i}, belonging to source RSU S_i with $x_{S_{ik}} \in X_{S_i}$ and $y_{S_{ik}} \in Y_{S_i} = \{0, 1\}$, a poisoning attack is defined as a targeted random label-flipping attack when the attacker randomly selects a fraction $\zeta \in (0, 1]$ of misbehaving vehicles and flips the label of the corresponding training samples from 1 to 0.*

In a policy induction attack, we properly adapt the adversarial attack originally presented in [12] in a DRL context. The malicious intent here is to force the target DRL agent to learn a policy selected by the adversary. We assume an exogenous attacker with minimal *a priori* information (e.g., input type and format) of the target DQN, and with knowledge of its reward function and the update frequency of the target network. The attacker can directly manipulate the target DQN's environment configuration, albeit with no control over the target network parameters, reward function, or optimization mechanism. According to Definition 2, the attacker creates

replica DQNs (i.e., Q', $\hat{Q}'$) of the target's DQN (i.e., Q, $\hat{Q}$) and initializes them with random parameters. Since the attacker has no knowledge of the target's DQN architecture and its parameters at every time step, a black-box technique is followed to exploit the transferability of adversarial examples. This is achieved by crafting state perturbations using replicas of the target's DQN such as those introduced in [13]. Moreover, the Fast Gradient Sign Method (FGSM)[2] algorithm is used to craft adversarial examples at every training time step.

Definition 2 (Policy induction attack) *The attacker induces an arbitrary adversarial policy π_{adv} on the target DQN by injecting adversarial examples into training data. Specifically, given state observation $\mathbf{s_{t+1}}$, the attacker crafts a perturbation vector ($\hat{\delta}_{\mathbf{t+1}}$) using FGSM at every training time-step and injects it into the target DQN's state space, as*

$$a'_{adv} \leftarrow \pi_{adv}(\mathbf{s_{t+1}}), \tag{5}$$

$$\hat{\delta}_{\mathbf{t+1}} \leftarrow Craft(\hat{Q}', a'_{adv}, \mathbf{s_{t+1}}), \tag{6}$$

$$\mathbf{s'_{t+1}} \leftarrow \mathbf{s_{t+1}} + \hat{\delta}_{\mathbf{t+1}}. \tag{7}$$

Crafting adversarial inputs $\mathbf{s'_t}$ and $\mathbf{s'_{t+1}}$ requires minimizing the loss function in (8) to force the target DQN to optimize towards action a', given state s_t, as

$$min_{\theta'}(y_t - Q'(\mathbf{s_t}, a_t; \theta'))^2, \tag{8}$$

where $y_t = r_t + \gamma max_{a'}\hat{Q}'(\mathbf{s'_{t+1}}, a'; \theta'_{-})$ and $Q^{'}$, $\hat{Q}'$ are the replica DQNs of the target DQN.

2.3 Transfer Learning-Based DRL Framework

The distributed RSU deployments in vehicular systems and the varying spatiotemporal behavior of traffic traces, render the training of a DRL-MDS agent difficult, especially in detecting unseen and partially observable misbehavior attacks. Challenges in such complex setups include slow convergence of the learning process, overfitting, and/or sub-optimal solutions due to poor exploration [15]. Hence, we advocate the adoption of a transfer learning-based DRL approach to achieve distributed collaborative misbehavior detection in adversarial vehicular environments. We rely on selectively transferring knowledge from trustworthy source RSUs to avoid negative knowledge sharing from adversary-influenced RSUs, as shown in Figure 2.

2.4 Transfer Learning for DRL-based Misbehavior Detection

TL leverages valuable knowledge acquired in one task and past experience to improve learning performance on other tasks (similar/dissimilar). The knowledge learned and previous experiences obtained from learning some *source* tasks, can be re-used to enhance the learning of some *target* tasks. In our work, the task represents the misbehavior detection performed in each RSU. Most DRL-based works in related literature (Section 2.1.1) have developed agents that can efficiently learn tabula rasa without any previously learned knowledge. Yet, tabula rasa DRL techniques require a long learning period to be effective, which can be inefficient in

[2]The FGSM method utilizes the neural network gradients' sign to create a perturbed adversarial input that maximizes the loss [14]. For a given input x, the adversarial input x' generated by the FGSM can be summarized as $x' = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$, where y is the label of input x, ϵ denotes a small multiplier, θ are the model parameters, and J represents the loss.

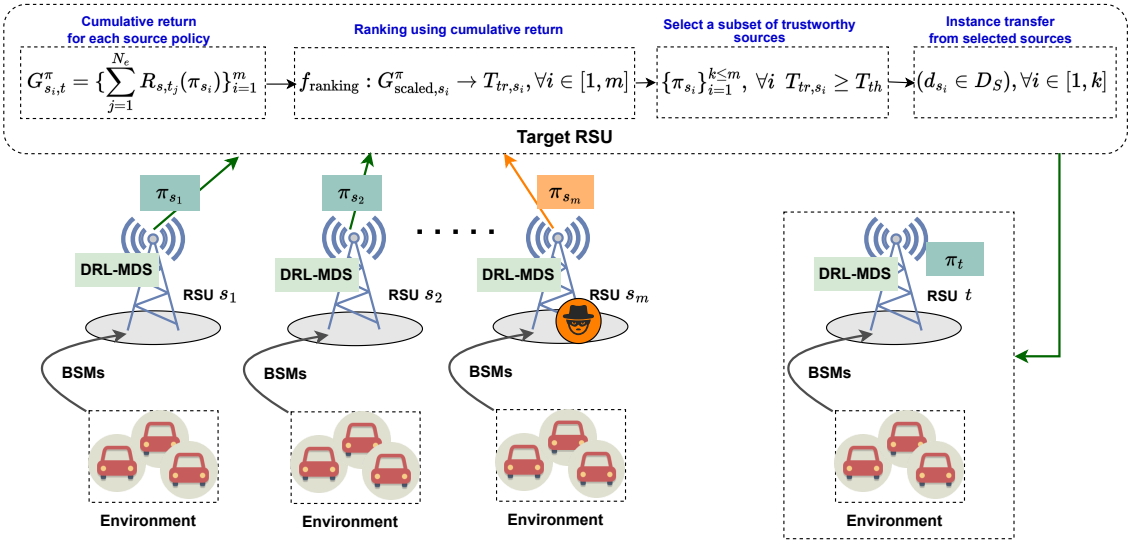

Figure 2: Visualization workflow for selecting trustworthy source RSUs. Source RSUs with policies $\{\pi_{s_i}\}_{i=1}^{m}$ are ranked (f_{ranking}) based on normalized scale cumulative return values $\{G^{\pi}_{\text{scaled},s}\}_{i=1}^{m} \in [0,1]$, which are extracted from the target RSU t's environment. Instances $\{(s_i^{(t)}, a_i^{(t)}, r_i^{(t)}, s_i^{(t+1)}) := d_{s_i}\}_{i=1}^{k}$ are transferred to the target RSU t from k subset of trustworthy source RSUs with policies $\{\pi_{s_i}\}_{i=1}^{k}$, where $k \leq m$ and D_S denotes samples of source RSUs. Green arrows denote positive transfer from trusted sources and the orange arrow represents a potential negative transfer from a malicious source [8].

computationally intensive and mission-critical operations [16]. For instance, an agent at the start of learning may tend to spend significant time exploring the environment before finding an optimal policy. Thus, TL can be leveraged to accelerate the learning. In this context, source and target RSUs associated with TL can be represented as MDP[3] agents. Accordingly, TL between RSUs can be defined as follows.

Definition 3 (Transfer learning between RSUs) *Given a set of m source RSUs with MDPs $\mathbf{M}_s = \{\bigcup_{i=1}^{m} \mathcal{M}_{i,s} | \mathcal{M}_{i,s} \in \mathbf{M}_s\}$ and a target RSU with MDP $\mathcal{M}_t$, the goal of TL is to learn an optimal policy π_t^* for the target RSU as*

$$\pi_t^* = \arg\max_{\pi_t} \mathbb{E}_{\mathbf{s}\sim\mathcal{S}_\mathbf{t}, a\sim\pi_t}[Q^{\pi_t}_{\mathcal{M}_t}(\mathbf{s}, a)], \tag{9}$$

by leveraging external information $\mathcal{I}_s$ from $\mathbf{M}_s$ along with internal information $\mathcal{I}_t$ from $\mathcal{M}_t$. In (9), $\pi_t = \phi(\mathcal{I}_s \sim \mathbf{M}_s, \mathcal{I}_t \sim \mathcal{M}_t) : \mathbf{s_t} \in \mathcal{S} \rightarrow a_t \in \mathcal{A}$ is a policy that maps states to actions for $\mathcal{M}_t$, which is learned from both $\mathcal{I}_t$ and $\mathcal{I}_s$. The policy π_t is estimated using a DNN.

In multi-source[4] RSU deployments, according to Definition 3, the aim is to enhance the learning of misbehavior detection at the target RSU by leveraging knowledge acquired from a set of source RSUs. Assume m source RSUs with MDPs $\mathbf{M}_s = \{(\mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{R}_i, \gamma_i)\}_{i=1}^{m}$,

[3]Henceforth, MDP and environment are used interchangeably.

[4]In a single-source scenario with $|\mathbf{M}_s| = 1$, tabula rasa learning occurs without any external transfer when $\mathcal{I}_s = \emptyset$.

and a detection policy π_{s_i} for each source RSU i. Then, the target RSU with MDP $\mathcal{M}_t = (\mathcal{S}_\mathbf{t}, \mathcal{A}_t, \mathcal{P}_\mathbf{t}, \mathcal{R}_t, \gamma_t)$ aims to enhance the learning of π_t^* by leveraging $\mathcal{I}_s \sim \mathbf{M}_s$. The external $\mathcal{I}_s$ represents transferred knowledge on misbehavior detection from source RSUs.

2.5 Knowledge Transfer

Previous works (e.g., [17, 18, 19]) on the integration of TL with collaborative DRL have demonstrated effectiveness in scenarios with some degree of similarity in agents' experiences. Based on the definition of transferred knowledge, TL techniques in DRL can be realized as policy[5], representation[6], and instance transfer [16, 20]. We hereby employ an instance TL technique to transfer misbehavior detection knowledge from a set of source RSUs to a target RSU. The rationale lies in transferring those source samples that can enhance the detection performance of the target RSU. Our method selects only related (i.e., good/useful) samples collected from the environments of source RSUs as agents' experiences. The selection of such samples prevents negative knowledge transfer that may originate from adversarial source RSUs. In collaborative DRL-MDS, positive transfer occurs when the transferred knowledge from source RSUs improves the target RSU's performance. In negative transfer, the target's performance degrades compared to tabula rasa learning.

To realize collaborative DRL-MDS, our proposed methodology employs TL with a new selective knowledge transfer scheme to prevent negative transfers from untrusted source RSUs, as in Figure 2. Under positive transfer, the target RSU is expected to achieve higher cumulative return as well as reach the asymptotic performance earlier than in tabula rasa learning. Hence, measuring the degree of similarity in misbehavior detection performed at the source and target RSUs becomes crucial to avoid negative transfer overhead in TL.

2.5.1 Trust Evaluation

In Figure 2, we provide a visualization workflow for trustworthy source RSUs' selection for transfer. We first establish an inter-agent semantic similarity metric between the source and target to ensure a positive transfer. A source RSU s_i with policy π_{s_i} is considered to have high *semantic relatedness* with the target RSU t, if π_{s_i} can generate the maximum cumulative return from the target RSU's environment in a limited number of training episodes. Inspired by works in [17, 21], the semantic relatedness between the source and target RSUs is defined as the gain of cumulative return for a policy π_{s_i} under the target reward function,

$$G_{s_i}^{\pi} = \sum_{j=1}^{N_e} R_{t_j}(\pi_{s_i}), \quad \forall i \in [1, m], \tag{10}$$

where N_e denotes the number of training episodes and $R_t(.)$ denotes the target reward function.

By utilizing semantic relatedness, the target RSU can effectively select a subset of source RSUs that are trustworthy to contribute towards enhanced misbehavior detection. A higher

[5] In policy transfer, a set of policies $\{\pi_{s_i}\}_{i=1}^m$ from source MDPs are transferred to the target MDP, and then the target policy π_t is learned by utilizing knowledge from them.

[6] In representation transfer, the algorithm learns a feature representation of the task, such as a value function $V^\pi(\mathbf{s})$ or the $Q^\pi(\mathbf{s}, a)$ function, and the knowledge learned can be either directly used in the target or indirectly using a task-invariant feature space.

cumulative return in N_e training episodes ascertains a positive transfer of detection knowledge and implies that a source RSU is more reliable for collaboration. The procedure of selecting trustworthy source RSUs for knowledge transfer is summarized in Algorithm 1. Without loss of generality, we normalize cumulative return $G^{\pi}_{s_i}$ in a way that the scaled cumulative return $G^{\pi}_{\text{scaled},s_i}$ resides in the range of $[0, 1]$.

Algorithm 1 Selection of trustworthy source RSUs for transfer

Input: Load $\pi_{s_1}, \pi_{s_2}, ..., \pi_{s_m}$, where $i \in [1, m]$
Output: Trusted policies $\pi_{s_1}, \pi_{s_2}, ..., \pi_{s_k}$, where $k \leq m$
1: **for** iteration $i = 1$ to m **do**
2: Compute $G^{\pi}_{s_i} = \sum_{j=1}^{N_e} R_{t_j}(\pi_{s_i})$
3: **end for**
4: $G^{\pi}_{\text{max}} = \text{max}(G^{\pi}_{s_1}, G^{\pi}_{s_2}, ..., G^{\pi}_{s_m})$
5: $G^{\pi}_{\text{min}} = \text{min}(G^{\pi}_{s_1}, G^{\pi}_{s_2}, ..., G^{\pi}_{s_m})$
6: $G^{\pi}_{\text{scaled},s_i} = \dfrac{G^{\pi}_{s_i} - G^{\pi}_{\text{min}}}{G^{\pi}_{\text{max}} - G^{\pi}_{\text{min}}}\ \forall i \in [1, m]$
7: $f_{\text{ranking}} : G^{\pi}_{\text{scaled},s_i} \rightarrow T_{tr,s_i}\ \forall i \in [1, m]$
8: Select k out of m sources with $T_{tr,s_i} \geq T_{th}$, where $T_{tr,s_i}, T_{th} \in [0, 1]$

In Algorithm 1, cumulative return is computed using (10) for each source RSU s_i with a policy π_{s_i} and Min-Max normalization is applied to scale cumulative returns $\{G^{\pi}_{s_i}\}_{i=1}^{m}$ into the range of [0,1]. Min-Max normalization based on cumulative return values is defined as

$$G^{\pi}_{\text{scaled},s_i} = \frac{G^{\pi}_{s_i} - G^{\pi}_{\text{min}}}{G^{\pi}_{\text{max}} - G^{\pi}_{\text{min}}}, \tag{11}$$

where G^{π}_{max} and G^{π}_{min} denote max$\{G^{\pi}_{s_i}\}_{i=1}^{m}$ and min$\{G^{\pi}_{s_i}\}_{i=1}^{m}$, respectively. As such, the scaled cumulative return $G^{\pi}_{\text{scaled},s_i}$ of a source RSU s_i can be associated with its trust value T_{tr,s_i}, which is used by the target RSU to decide whether the source is trustworthy to collaborate with. We introduce a source ranking strategy f_{ranking} in Algorithm 1 (line 7), used by the target RSU to sort source RSUs based on their trust values and select a subset lying above a specified threshold value $T_{th} \in [0, 1]$. The selection of a threshold to ascertain the trustworthiness of source RSUs can be either tolerant with a lower T_{th} value or more stringent with a higher T_{th} value. Reputation or trust-based methods in related literature typically use a $[0, 1]$ scale to represent trust/reputation values and consider 0.5 as a neutral value or a threshold [22, 23].

2.5.2 Selective Knowledge Transfer

Upon the completion of source RSUs' selection, the target RSU applies instance TL by collecting samples from k trustworthy source RSUs following their policies $\{\pi_{s_i}\}_{i=1}^{k}$ (i.e., the output of Algorithm 1). In target RSU training, we employ a selective knowledge transfer scheme, called *experience selection*, that selects source samples with high semantic relatedness. Motivated by [17, 24], experience selection is defined as

$$Q^*(\mathbf{s}, a) \geq y > Q_{\theta}(\mathbf{s}, a), \tag{12}$$

where $Q^*(.)$ represents an optimal Q-function and $y = R_t(\mathbf{s}, a) + \gamma \text{max}_{a_{t+1}} Q_{\theta}(\mathbf{s_{t+1}}, a_{t+1})$. The aim of the target RSU is to learn the optimal misbehavior detection policy π^*_t by obtaining an

optimal Q-function $Q^*(\mathbf{s}, a)$ for a given state-action pair. The expected return achieved by following an optimal policy is always greater or equal to that of any arbitrary behavior policy μ; hence, the expected return $Q_\mu(\mathbf{s}, a)$ of any arbitrary behavior policy μ can serve as a lower bound of the optimal Q-value $Q^*(\mathbf{s}, a)$. Lower bound Q-learning [24] can be expressed as

$$Q^*(\mathbf{s}, a) \geq Q_\mu(\mathbf{s}, a) = \mathbb{E}_\mu[r_t + \sum_{k=t+1}^{T} \gamma^{k-t} r_k], \tag{13}$$

with convergence guarantees of Q-values [25]. The lower bound Q-value in (13) implies that the estimated $Q_\theta(\mathbf{s}, a)$ value in (12) is lower than the optimal $Q^*(\mathbf{s}, a)$ value. Following lower bound Q-learning, experience selection aims to update $Q_\theta(\mathbf{s}, a)$ towards the lower bound of $Q^*(\mathbf{s}, a)$ with source samples of high semantic relatedness, i.e., $Q_\theta(\mathbf{s}, a) \geq y$.

Algorithm 2 Target RSU training with transferred knowledge

Input: Load $\pi_{s_1}, \pi_{s_2}, ..., \pi_{s_k}$ (output of Algorithm 1);
Initialize experience samples buffer $\tilde{\mathcal{S}}$, replay memory buffer $\mathcal{D}$, action-value function Q with random weights θ and discount factor γ

1: **for** episode = 1, M **do**
2: Initialize state sequence $\mathbf{s_1}$
3: **for** $t = 1$, T **do**
4: Pick a random value $rnd \in (0, 1)$
5: **if** $\epsilon > rnd$ **then**
6: Select a random action $a_t \in \mathcal{A}$ with probability ϵ
7: **else**
8: Select $a_t \in \mathcal{A}$ following policy π_t
9: **end if**
10: Execute a_t, and observe reward r_t and next state $\mathbf{s_{t+1}}$
11: Collect samples $\mathcal{D} = \{(\mathbf{s_t}, a_t, \mathbf{s_{t+1}}, r_t)\}$ using π_t
12: Collect samples $\tilde{\mathcal{S}} = \{(\tilde{\mathbf{s}}_\mathbf{t}, \tilde{a}_t, \tilde{\mathbf{s}}_\mathbf{t+1}, \tilde{r}_t)\}$ using $\pi_{s_i} \forall i \in [1, k]$
13: $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup \mathcal{D}$
14: Sample random minibatch of transitions $(\mathbf{s_t}, a_t, \mathbf{s_{t+1}}, r_t)$ from $\tilde{\mathcal{S}}$
15: Set $y_t = \begin{cases} r_t, & \text{if } \mathbf{s_{t+1}} \text{ terminal} \\ r(\mathbf{s_t}, a_t) + \gamma \max_{a_{t+1}} Q_\theta(\mathbf{s_{t+1}}, a_{t+1}), & \text{otherwise} \end{cases}$
16: **if** $Q_\theta(\mathbf{s_t}, a_t) \geq y_t$ **then**
17: Select corresponding samples for target model update
18: **else**
19: Remove corresponding transitions $(\tilde{\mathbf{s}}_\mathbf{t}, \tilde{a}_t, \tilde{\mathbf{s}}_\mathbf{t+1}, \tilde{r}_t)$ from $\tilde{\mathcal{S}}$
20: **end if**
21: Gradient descent on $(Q_\theta(\mathbf{s_t}, a_t) - y_t)^2$ according to (15)
22: **end for**
23: **end for**

The target RSU training with selective knowledge transfer is summarized in Algorithm 2. The algorithm proceeds as follows to improve the learning performance of the target RSU with the aid of experience selection. The training of the target RSU is divided into episodes. In each time step, the target RSU selects an action $a_t \in \mathcal{A}$ either randomly with probability ϵ or according to the best Q-value given by (3) (lines 4-9). Subsequently, the chosen action a_t is executed in the environment, and the reward r_t and the next state $\mathbf{s_{t+1}}$ are observed (line 10). Assume k trustworthy source RSUs with policies $\{\pi_{s_i}\}_{i=1}^k$. The target RSU can form an experience samples buffer $\tilde{\mathcal{S}}$ by collecting samples following k source policies (line 12) and

computes rewards using its current reward function. The amount of samples collected into $\tilde{\mathcal{S}}$ from each trustworthy source RSU s_i is equivalent to $\eta_{s_i} \cdot |\tilde{\mathcal{S}}|$, with

$$\eta_{s_i} = \frac{T_{tr,s_i}}{\sum_{i=1}^{k} T_{tr,s_i}}, \quad \forall i \in [1,k], \tag{14}$$

where $T_{tr,s_i} = G^{\pi}_{\text{scaled},s_i}$ in (11). Consequently, source RSUs with higher trust values will transfer more samples, which results in learning more from a set of highly trusted RSUs. To trade off exploitation with exploration, target samples are also added to $\tilde{\mathcal{S}}$ following the online Q-network with ϵ-greedy policy (line 13). In the target's model update, parameterized by θ, training samples are drawn from $\tilde{\mathcal{S}}$ (line 14), and experience selection is applied (lines 16-17) to filter out relevant samples. Conversely, samples that are not semantically important will be removed from $\tilde{\mathcal{S}}$ (lines 18-19). As such, $\tilde{\mathcal{S}}$ is gradually updated while equally prioritizing the remaining training samples, which results in improved sample efficiency. The target RSU can thus leverage selective knowledge transfer and train its misbehavior detection model by minimizing the loss function,

$$L(\theta) = \frac{1}{2} \sum_{t} \left\| Q_\theta(\mathbf{s_t}, a_t) - y_t \right\|^2, \tag{15}$$

where target Q-values are given by $y_t = R(\mathbf{s_t}, a_t) + \gamma \text{max}_{a_{t+1}} Q_\theta(\mathbf{s_{t+1}}, a_{t+1})$. Moreover, the selection of samples with $Q_\theta(\mathbf{s_t}, a_t) \geq y_t$ implies that learning updates in (15) are encouraged towards the lower bound of the optimal $Q^*(\mathbf{s}, a)$ value. Consequently, experience selection updates the misbehavior detection policy π_t towards the optimal policy π_t^*.

Although experience selection improves sample efficiency, transferred knowledge from source instances may introduce bias owing to differences in data distribution or variability in misbehavior attack patterns. Therefore, to circumvent such bias, experience selection is utilized only in early training stages and, subsequently, target RSU training switches to traditional DQN learning. Algorithm 2 for traditional DQN operates similarly but without sampling from $\tilde{\mathcal{S}}$ (lines 13-14) and with no experience selection (lines 16-20). In this case, the target RSU samples a minibatch of transitions from replay memory buffer $\mathcal{D}$ with training samples from the target environment, and those samples are directly used to train the misbehavior detection model by minimizing (15).

2.6 Experiments

Leveraging the open-source VeReMi dataset [9], we conduct experiments on a set of simulation scenarios, listed in Table 1, to assess the proposed knowledge transfer approach. Our goal is to provide empirical evidence of the benefits of TL through simulated scenarios for collaborative misbehavior detection in unpredictable and untrusted vehicular environments. The considered scenarios aim to cover diversified aspects of collaborative misbehavior detection by means of knowledge transfer between RSUs. Vehicular mobility renders indispensable the study of such scenarios that may arise in a collaborative misbehavior detection setup. With that aim, we exploit the attack variants present in VeReMi. For each specific scenario, a different set of misbehavior attacks is selected, aiming to provide sufficient coverage of the available attacks in VeReMi.

Scenario	Objective
SC1	Detect future misbehaviors of the same type
SC2	Detect unseen/unknown misbehaviors of similar type
SC3	Detect partially observable misbehaviors

Table 1: Examined Scenarios

2.6.1 Examined Scenarios

We conduct experiments on a set of simulation scenarios, listed in Table 1, to assess the proposed knowledge transfer approach. The goal is to provide empirical evidence of the benefits of TL through simulated scenarios for collaborative misbehavior detection in unpredictable and untrusted vehicular environments. The considered scenarios aim to cover diversified aspects of collaborative misbehavior detection by means of knowledge transfer between RSUs. In what follows, we elaborate on the considered scenarios.

SC1: The first scenario captures the requirement of detecting future misbehaviors of the same type. In SC1, a misbehaving vehicle can permeate an attack across multiple geographic locations across its trajectory, resulting in some RSUs experiencing the attack earlier in time (sources) compared to others (targets). SC1 arises in situations where RSU deployments are sparsely distributed in vehicular setups. In this case, the target RSU relies on collaborative misbehavior detection to proactively detect future attacks of the same type.

SC2: In this scenario, the target RSU aims to acquire knowledge from source RSUs to detect an unseen/unknown attack. Such situations may be encountered in practice due to blind spots and occlusions caused by vehicular infrastructure or moving vehicles. This inevitably results in limited situational awareness in some RSUs (targets) which may not experience certain attack types. In this case, the target RSU seeks relevant knowledge from source RSUs with expertise in detecting attacks of similar type. We assume that source RSUs are capable of detecting a wider range of misbehavior attack types. On the contrary, the target RSU is trained to identify a narrower set of attacks compared to source RSUs. In such cases, the target RSU aims to leverage knowledge transfer from source RSUs to identify similar new misbehavior attacks.

SC3: In this scenario, source RSUs are trained with different feature vectors resulting in partial observability of the attack space. Such cases arise in practice when RSUs have diverse computational capabilities, including different processing and buffer sizes, limiting their training only on specific attack types. This heterogeneity may hinder an RSU's ability to process multiple/high-dimensional feature vectors effectively. Thus, the target RSU needs to select source RSUs based on relevant feature vectors to detect an array of misbehaviors and enhance the effectiveness of collaborative detection.

2.7 Performance Evaluation

This section presents the learning performance of DRL-MDS agents for the scenarios described above using VeReMi. Next, the resulting detection performance achieved via knowledge transfer is assessed for different misbehavior attack types.

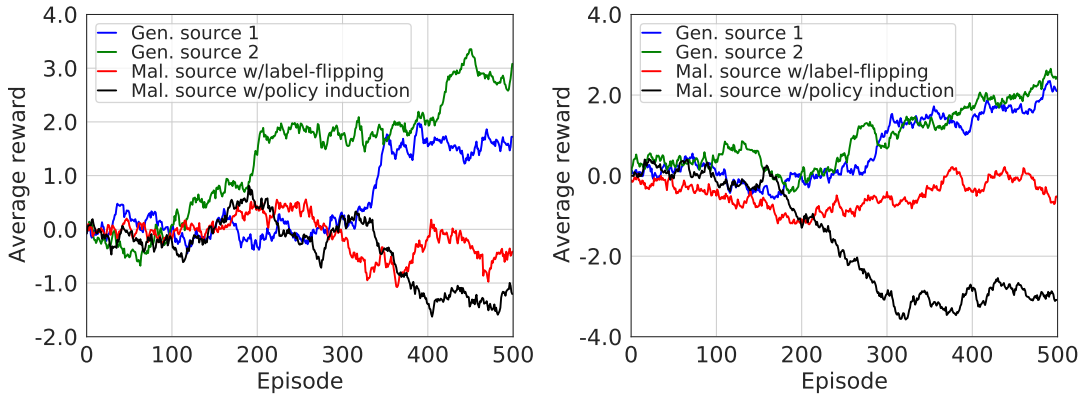

Figure 3: Learning performance of source RSUs in SC1 for (Left) Random Position and (Right) Random Position Offset misbehavior attack types.

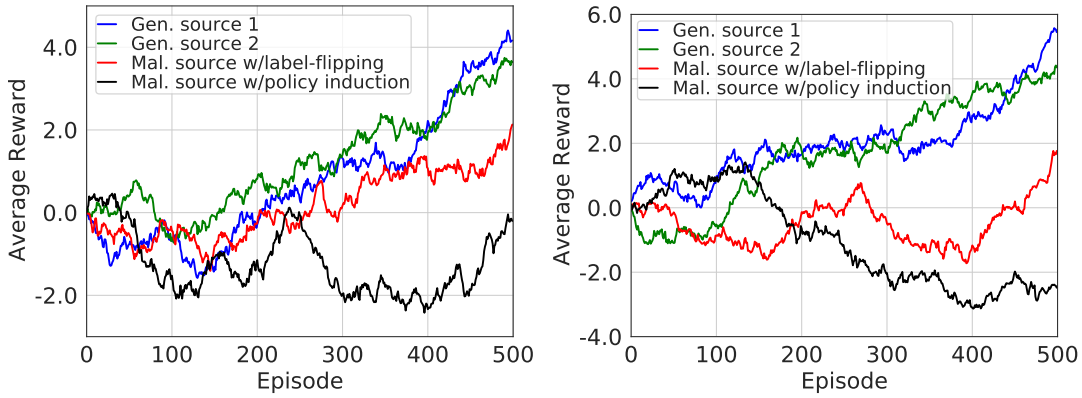

Figure 4: Learning performance of source RSUs in SC2 for (Left) a combination of DoS, DoS Random and DoS Random Sybil; (Right) a combination of DoS, DoS Disruptive and DoS Disruptive Sybil misbehavior attack types.

2.7.1 Learning Performance

We evaluate the learning performance of misbehavior detection agents in *i*) the source RSUs with trained policies and *ii*) the target RSU with and without knowledge transfer.

Source RSUs

To assess the contribution of source RSUs towards knowledge transfer, we measure the average reward gains per RSU during the interactions with their own environments.

Figure 3 illustrates the learning performance of source RSUs for two position-related misbehavior types considered for SC1. As can be visually comprehended, genuine source RSUs accumulate significantly higher average rewards compared to malicious source RSUs, which are detrimentally influenced by label-flipping and policy induction attacks. Furthermore, the negative impact on the victim source RSU from the policy induction attack is greater than that of the label-flipping attack. This can be attributed to the effectiveness of the policy induction

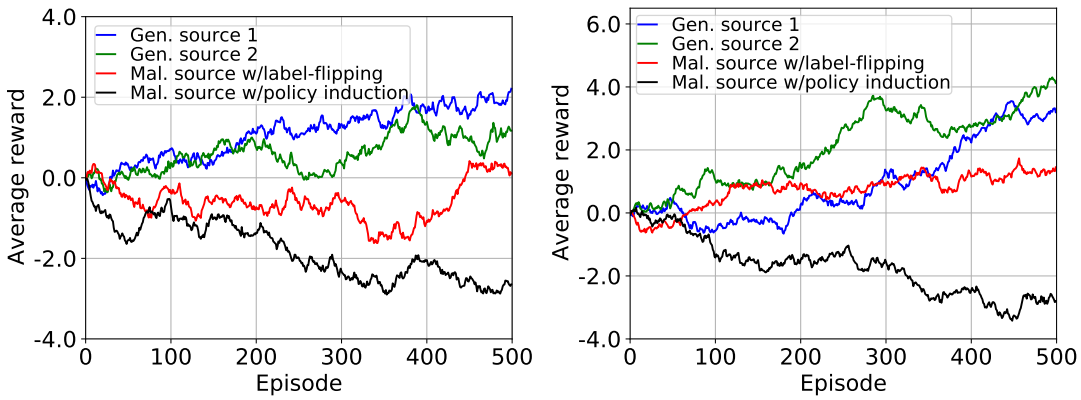

Figure 5: Learning performance of source RSUs in SC3 for (Left) position variants with Constant Position Offset (Gen. source 1), Random Position (Gen. source 2), and Random Position Offset (Mal. sources) misbehaviors; (Right) speed variants with Constant Speed Offset (Gen. source 1), Random Speed (Gen. source 2), and Random Speed Offset (Mal. sources) misbehaviors.

attack which injects crafted inputs at each training step towards the adversary's goal. This, in turn, results in misclassifying malicious samples as genuine ones with increased false alarms. Similar behavior can be observed in Figure 4 for DoS-related misbehavior variants considered in SC2. In this case, each source RSU is trained on a combination of misbehavior attack types, providing the capability to detect additional misbehavior attack types compared to SC1. In Figure 5, the learning performance of source RSUs in SC3 for both position- and speed-related misbehavior variants is shown. Similarly to SC1 and SC2, genuine sources perform significantly better than malicious ones, while the pernicious impact of policy induction attacks compared to label-flipping becomes apparent.

Target RSU

As shown in Algorithm 1, based on the available source policies, the target RSU selects a subset of source RSUs using T_{th} to realize knowledge transfer. The proposed trust evaluation supports dynamic thresholding to handle unpredictable and untrusted vehicular environments, such as the erratic behavior of RSUs due to adversarial attacks. In our case, we select 0.5 and 0.8 for T_{th} to assess the impact of different threshold values on knowledge transfer without loss of generality.

Figures 6, 7, and 8 depict the learning performance of the target RSU in SC1, SC2, and SC3, respectively. We report average rewards for knowledge transfers involving different sources by calculating the average across three runs. The case of no transfer between source and target RSUs (i.e., tabula rasa learning) is also evaluated as a baseline. In addition, the black dashed lines in Figures 6–8 represent the best return reward of the baseline scheme with tabula rasa learning. These lines demonstrate the training time reduction achieved under each transfer at the target RSU to reach a certain performance level. Moreover, Table 2 presents specific information regarding the reduction in training time achieved through knowledge transfer, detailing the required number of training episodes and the corresponding wall-clock time.

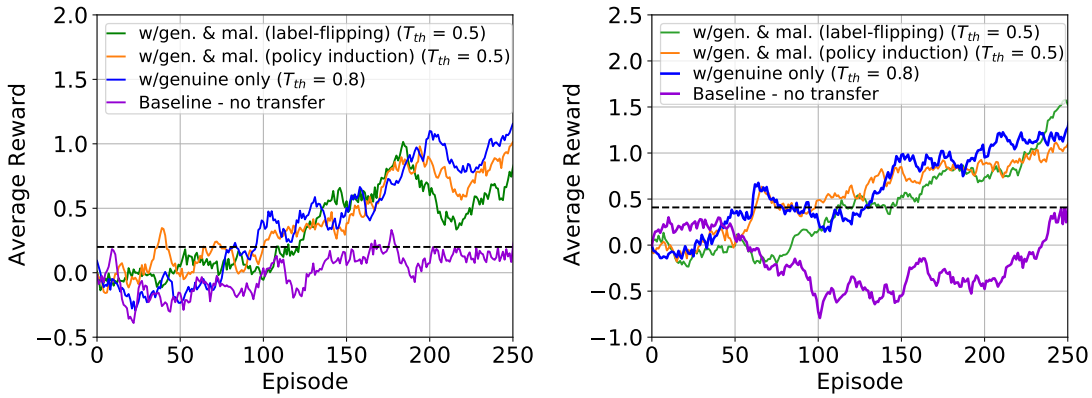

Figure 6: Learning performance of the target RSU, averaged across three runs, in SC1 for (Left) Random Position and (Right) Random Position Offset misbehavior attack types.

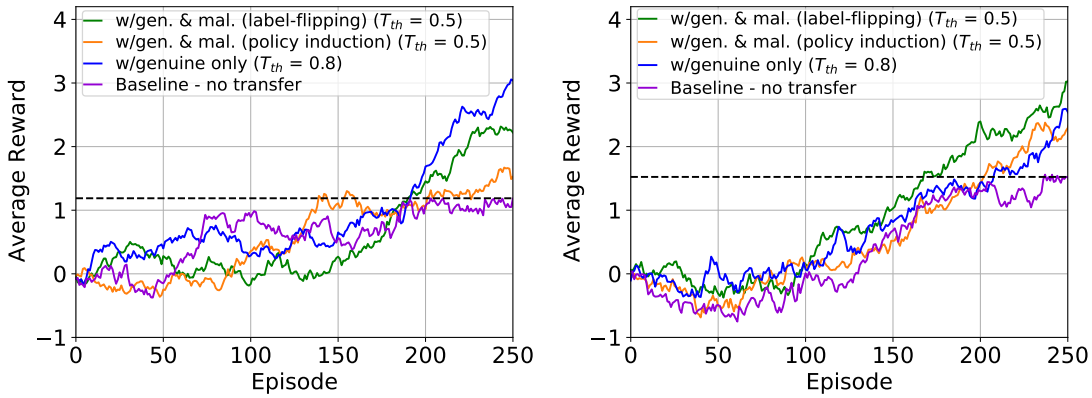

Figure 7: Learning performance of the target RSU, averaged across three runs, in SC2 for (Left) a combination of DoS and DoS Random; (Right) a combination of DoS and DoS Disruptive misbehavior attack types.

Figure 6 illustrates average rewards accumulated by the target RSU for a specific misbehavior with knowledge transferred from the source RSUs (shown in Figure 3). It can be observed that our selective knowledge transfer approach significantly enhances the learning performance compared to tabula rasa. Specifically, the target RSU achieves a higher cumulative return and reaches asymptotic performance earlier than in the baseline. In addition, Figure 6 reveals the impact of different threshold values on learning performance. When a tolerant $T_{th} = 0.5$ value is set, both genuine and malicious (label-flipping and policy induction) source RSUs feature in the transfer. With a stringent $T_{th} = 0.8$, collaboration stems only from genuine source RSUs. Interestingly, even though malicious source RSUs are involved with $T_{th} = 0.5$, the sample contribution in (14) and the experience selection in (12) ascertain positive transfer, resulting in similar performance as in $T_{th} = 0.8$.

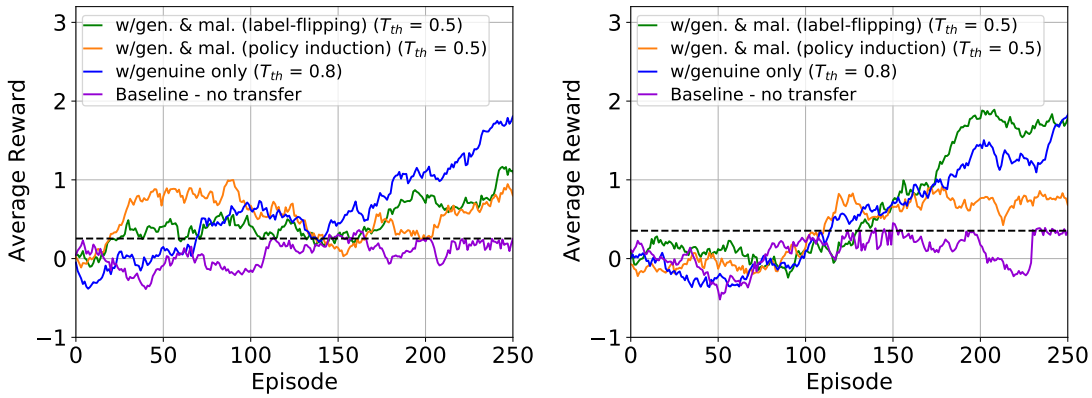

Figure 8: Learning performance of the target RSU, averaged across three runs, in SC3 for (Left) position-related and (Right) speed-related misbehavior variants.

Examined Scenario	Misbehavior Type(s)	Target Return	Training time in episodes (reduction %)				Training time in wall-clock time (mins) (reduction %)			
			Baseline	TF[1]	TF[2]	TF[3]	Baseline	TF[1]	TF[2]	TF[3]
SC1	Random Position	0.2005	250	122 (51%) (best return: 0.8173)	100 (60%) (best return: 1.0156)	96 (62%) (best return: 1.1575)	83.32	31.33 (62%)	26.68 (68%)	24.35 (71%)
	Random Position Offset	0.4092	250	144 (42%) (best return: 1.5233)	98 (61%) (best return: 1.0960)	128 (49%) (best return: 1.1574)	82.93	36.83 (56%)	26.14 (68%)	32.02 (61%)
SC2	DoS Random Sybil	1.1878	250	190 (24%) (best return: 2.2230)	201 (20%) (best return: 1.5293)	192 (23%) (best return: 3.0571)	84.57	50.16 (41%)	52.29 (38%)	50.68 (40%)
	DoS Disruptive Sybil	1.5235	250	168 (33%) (best return: 3.0216)	202 (19%) (best return: 2.3775)	208 (17%) (best return: 2.5939)	85.89	44.23 (48%)	53.12 (38%)	55.14 (36%)
SC3	Position variants*	0.2548	250	162 (35%) (best return: 1.1656)	166 (34%) (best return: 0.7982)	142 (43%) (best return: 1.8103)	79.18	39.37 (50%)	40.35 (49%)	36.20 (54%)
	Speed variants**	0.3541	250	130 (48%) (best return: 1.7811)	111 (55%) (best return: 0.6789)	114 (54%) (best return: 1.8242)	80.59	32.19 (60%)	27.49 (66%)	28.65 (64%)

TF[1]: w/genuine+malicious (label-flipping) ($T_{th} = 0.5$) TF[2]: w/genuine+malicious (policy induction) ($T_{th} = 0.5$)
TF[3]: w/genuine only ($T_{th} = 0.8$)
* Constant Position Offset, Random Position, and Random Position Offset
** Constant Speed Offset, Random Speed, and Random Speed Offset

Table 2: Summary of training time and the best return reward at the target RSU under knowledge transfer

2.7.2 Detection Performance

Detection performance is assessed in all scenarios in terms of *Accuracy* (A), *Precision* (P), *Recall* (R), and *F-score* (F),

$$\mathrm{A} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{16}$$

$$\mathrm{P} = \frac{TP}{TP + FP}, \tag{17}$$

$$\mathrm{R} = \frac{TP}{TP + FN}, \tag{18}$$

$$\mathrm{F} = 2\frac{RP}{R + P}, \tag{19}$$

respectively, by considering both genuine and misbehavior classes for each misbehavior type in VeReMi. The accuracy in (16) indicates the ratio of all correct predictions to the total number of considered input samples. The higher precision values in (17) indicate low FP rates, whereas higher recall values in (18) indicate low FN rates. The F-score in (19) provides a harmonic mean between precision and recall, which is used when FPs and FNs are vital. Therefore, a

Examined Scenario	Misbehavior Type(s)	Knowledge Transfer	Accuracy	Precision	Recall	F-score
SC1	Random Position	w/genuine+malicious (label-flipping) ($T_{th} = 0.5$)	0.9928	0.9778	0.9983	0.9879
		w/genuine+malicious (policy induction) ($T_{th} = 0.5$)	0.9933	0.9789	0.9987	0.9887
		w/genuine only ($T_{th} = 0.8$)	0.9920	0.9756	0.9977	0.9865
		Baseline (no transfer)	0.9497	0.8978	0.9362	0.9166
	Random Position Offset	w/genuine+malicious (label-flipping) ($T_{th} = 0.5$)	0.9896	0.9721	0.9933	0.9826
		w/genuine+malicious (policy induction) ($T_{th} = 0.5$)	0.9918	0.9764	0.9963	0.9862
		w/genuine only ($T_{th} = 0.8$)	0.9908	0.9730	0.9966	0.9847
		Baseline (no transfer)	0.9474	0.8944	0.9317	0.9127
SC2	DoS Random Sybil	w/genuine+malicious (label-flipping) ($T_{th} = 0.5$)	0.5659	0.5616	0.9915	0.7144
		w/genuine+malicious (policy induction) ($T_{th} = 0.5$)	0.5690	0.5606	0.9873	0.7150
		w/genuine only ($T_{th} = 0.8$)	0.5656	0.5581	0.9929	0.7145
		Baseline (no transfer)	0.5078	0.4730	0.5108	0.4912
	DoS Disruptive Sybil	w/genuine+malicious (label-flipping) ($T_{th} = 0.5$)	0.5544	0.5552	0.9955	0.7129
		w/genuine+malicious (policy induction) ($T_{th} = 0.5$)	0.5523	0.5544	0.9898	0.7107
		w/genuine ($T_{th} = 0.8$)	0.5552	0.5557	0.9960	0.7134
		Baseline (no transfer)	0.4928	0.4587	0.5018	0.4793
SC3	Position variants*	w/genuine+malicious (label-flipping) ($T_{th} = 0.5$)	0.9224	0.7982	0.9856	0.8822
		w/genuine+malicious (policy induction) ($T_{th} = 0.5$)	0.9195	0.7956	0.9779	0.8774
		w/genuine only ($T_{th} = 0.8$)	0.9236	0.7998	0.9880	0.8840
		Baseline (no transfer)	0.4978	0.2956	0.5095	0.3741
	Speed variants**	w/genuine+malicious (label-flipping) ($T_{th} = 0.5$)	0.9447	0.8540	0.9824	0.9135
		w/genuine+malicious (policy induction) ($T_{th} = 0.5$)	0.9348	0.8275	0.9857	0.8997
		w/genuine only ($T_{th} = 0.8$)	0.9394	0.8399	0.9845	0.9062
		Baseline (no transfer)	0.4988	0.2948	0.4953	0.3696

*Constant Position Offset, Random Position, and Random Position Offset **Constant Speed Offset, Random Speed, and Random Speed Offset

Table 3: Performance comparison for collaborative misbehavior detection in each scenario

higher F-score implies better performance in our examined scenarios.

Table 3 presents the performance results obtained from our comprehensive analysis of collaborative misbehavior detection with the selective knowledge transfer approach. Results show that misbehavior detection using knowledge transfer yields high effectiveness with a very high F-score of 0.98 in SC1 under both T_{th} values of 0.5 and 0.8, while correctly identifying misbehaviors with low rates of FPs and FNs. It should be highlighted that the baseline scheme with tabula rasa also achieves a significantly high F-score of 0.91. This is due to the fact that tabula rasa learning obtains sufficient knowledge during training to detect future misbehaviors of the same type. Although the results are reported only for two misbehavior types in SC1, similar performance levels were observed for other misbehavior types in VeReMi.

Numerical results for SC2 demonstrate effective identification of unseen attacks with knowledge transfer, achieving an F-score of 0.71 under both T_{th} values. Additionally, recall values approaching 1.0 further elucidate that such non-anticipated attacks can be successfully detected with a very low rate of FNs. As shown in Table 3, the baseline scheme is ineffective in detecting unseen attacks and achieves very low F-scores of 0.49 and 0.48 in contrast to knowledge transfers when encountering DoS Random Sybil and DoS Disruptive Sybil misbehaviors, respectively. This validates that the target RSU enhances its situational awareness and detects non-anticipated misbehaviors by acquiring knowledge from source RSUs.

Detection performance in Table 3 reveals that target learning with knowledge transfer yields significantly superior F-scores compared to the baseline scheme in SC3. Under both T_{th} values, the F-scores of 0.88 and slightly over 0.90 for position- and speed-related misbehaviors, respectively, demonstrate high effectiveness in detecting a partially observable attack space. Conversely, the baseline scheme becomes highly ineffective, as shown by the low F-score of 0.37 with a high number of FPs and FNs for both position- and speed-related misbehaviors. It is worth noting that misbehavior variants generated by adding/subtracting an offset (i.e.,

Constant/Random Position Offset and Constant/Random Speed Offset) are more challenging to detect as compared to others, such as Constant Position and Constant Speed. Thus, the transfer of relevant knowledge to the target RSU becomes imperative to effectively identify such partially observable attack spaces.

Overall, across all three scenarios, collaborative misbehavior detection with knowledge transfer significantly outperforms tabula rasa learning, as summarized in Table 3, demonstrating its high effectiveness. Specifically, our approach enhances robustness and generalizability by effectively detecting previously unseen and partially observable misbehavior attacks.

3 ZSM-based Security Slice Management for DDoS Attack Protection in Edge-based V2X Environments

3.1 Introduction

3.1.1 Background

The heterogeneity of beyond-5G (B5G) networks, combined with the dynamic nature of vehicular environments, necessitates robust, automated, and intelligent security mechanisms. Emerging V2X use cases are becoming especially complex given the ubiquitous mobility and criticality of the associated communication and services. The deployment of such on-demand services heavily relies on edge computing capabilities to fulfill the stringent V2X requirements in terms of latency and throughput. In this context, edge nodes can directly benefit from autonomous dynamic management capabilities at the edge, which (i) allocate dedicated resources to ensure the correct deployment and operation of services; (ii) migrate these services seamlessly and with anticipation; (iii) offload the computational load (e.g., delegate computationally intensive functions) [26].

Inevitably, security becomes one of the major concerns, due to the inherent V2X vulnerabilities and breaches, with multi-faceted threat vectors which an adversary may maliciously exploit to intrude the system. On top of this, decentralized edge deployments render the attack surface sufficiently large and may further exacerbate the V2X security risks. Among various attack types, Denial-of-Service (DoS) attacks constitute one of the salient threats against edge infrastructures hosting V2X services [27]. In DoS attacks, an attacker tries to prevent legitimate users from accessing the network and services, causing traffic disruption which may destabilize the V2X system and threaten user safety. DoS attackers typically flood the network either with traffic of higher frequency than the system can handle or with high computational requests, resulting in an overload of computational resources. This inevitably causes extensive periods of service unavailability where legitimate users cannot be served. When such attacks are launched from multiple sources, often in spatially distant locations, this results in distributed DoS (DDoS) attack variants [28].

Current technologies that cooperate to offer advanced services with an end-to-end (E2E) perspective, could also jointly serve to provide protection against certain DDoS attacks. However, for protecting the entire V2X attack surface, their potential becomes limited without intelligent orchestration engines. The orchestration layer is an essential element in ensuring system automation, abstracting the complexity of the coordination and management of technologies and domains (i.e., RAN, edge, cloud). Moreover, when orchestration processes are combined with (i) policy-based approaches, that provide flexibility in specifying requirements, and (ii) intelligent decision engines, the benefits of abstracting the complexity of the underlying technologies are highly leveraged, enabling B5G V2X use cases with highly diverse requirements in terms of QoS and security [29]. In line with policy-based approaches, it is of utmost importance to ensure that the right security level is properly applied based on the user requirements. Security Service-Level Agreements (SSLAs) [30] are thus employed for this aim, serving as a contract between the user and the operator.

Recent advances in the areas of 5G, V2X communication and security are also driven by consistent standardization activities from relevant organizations (e.g., 3GPP, ITU, ETSI, and

IEEE). Such standardization bodies also define specifications for key enabling technologies, such as Zero-touch network and Service Management (ZSM), E2E slicing, and edge computing, to enhance automation and security in future networks [31][32]. In particular, ZSM represents a visionary next-generation management system with the ultimate goal of achieving complete automation in all operational processes and tasks. These tasks include planning and design, delivery, deployment, provisioning, monitoring, and optimization, all of which are ideally executed without human intervention [33]. In addition, a growing number of standards-developing organizations steer their efforts towards integrating data-empowered solutions for V2X security. Over the next years, network operators are also expected to advance the implementation of automated E2E slicing management and security enforcement functionalities for V2X systems by adopting standards such as ZSM [34].

3.1.2 Contribution

This section introduces a novel framework that combines the ZSM paradigm with E2E security slicing to provide adaptive, policy-driven protection against DDoS threats [35]. In particular, we demonstrate the feasibility of a ZSM-based framework to autonomously protect V2X services located at the edge in a B5G infrastructure by effectively mitigating various DDoS attack types. The threefold contribution of this article can be summarized as follows:

- We perform autonomous and ZSM-compliant security management of a real B5G network and services through the deployment and enforcement of E2E B5G security slices. The management involves the dynamic reconfiguration of B5G components (i.e., RAN and V2X aggregator) in order to fulfill the SSLA.
- We incorporate security as a *native* element in the E2E ETSI slice management standard to autonomously manage the E2E B5G security slices, including the E2E logical coordination of different domains to deploy and interconnect per-domain security sub-slices.
- We utilize the V2X aggregator and gNBs as security enforcement points to mitigate DDoS attacks at two levels; firstly, where the attack is detected and, eventually, as close as possible to the source. Adversaries are banned from the domain under attack, while the mitigation countermeasure is shared with neighboring domains to avoid the propagation of malicious information.

To showcase the detection capabilities of our approach at the edge, we integrate a data-driven DDoS attack detection methodology, originally introduced in [10], based on DRL. By performing experiments using an open-source dataset, our framework is shown to be highly effective in detecting various DDoS attack variants while keeping detection latency at low levels. The subsequent subsections present an overview of the proposed framework, with further details available in [35].

3.2 ZSM-Based Security Slice Management Framework

The proposed framework is based on the ETSI ZSM reference architecture and is designed to enable full lifecycle management of E2E security slices in V2X scenarios. It introduces both proactive and reactive security mechanisms through closed-loop automation. By using this

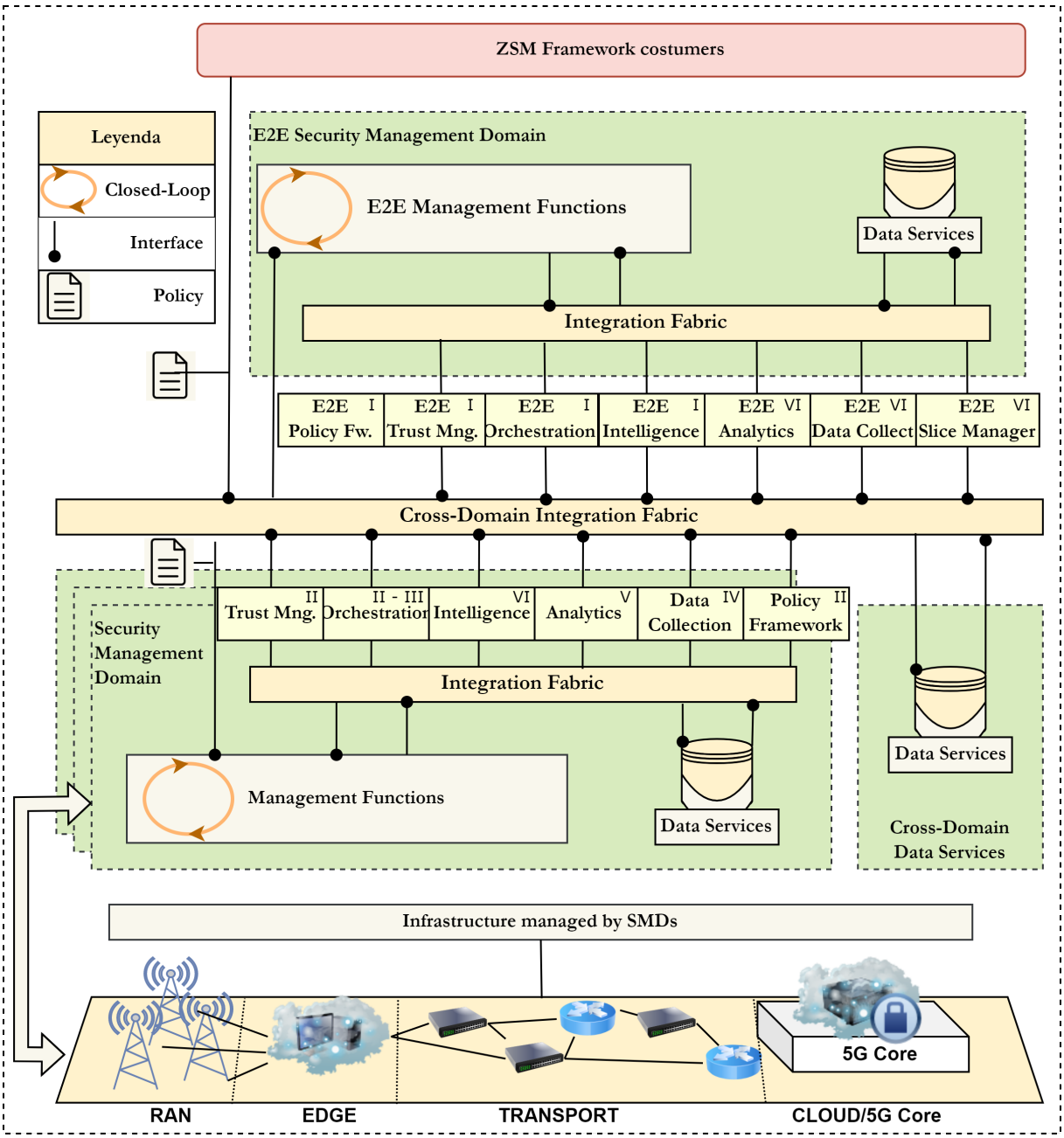

Figure 9: ZSM-based architecture of the proposed framework [35]. The Roman numerals on the top right of each entity of the architecture denote the step of the closed loop in which they participate.

standardized reference architecture, our security framework not only accelerates the deployment of innovative V2X services but also enhances operational efficiency, minimizes downtime, and improves end-user experience.

As shown in Figure 9, the framework consists of an E2E Security Management Domain (SMD) and as many independent SMDs as required to manage the underlying physical infrastructure. Although the design is hierarchical, each of these independent SMDs is capable of autonomously managing resources through closed-loop processes such as self-* (observation,

analysis, and decision-making). The E2E SMD performs the synchronization of multi-domain tasks [36].

3.2.1 Closed-Loop Management

The closed-loop management cycle consists of proactive phases (e.g., initial deployment, configuration) and reactive phases (e.g., anomaly detection, dynamic reconfiguration). The proactive phase interprets service and security requirements defined in Security Service Level Agreements (SSLAs) and Network Slice Templates (NSTs). These are converted into Medium-level Security Policy Language for Orchestration (MSPL-OP) policies, which are used to configure security assets across domains. The reactive phase involves real-time monitoring and analytics. If SSLA compliance is violated or a threat is detected, the system dynamically generates mitigation policies and applies them through orchestration mechanisms in affected domains [36].

3.2.2 Security Enablers

The framework employs a variety of specialized security enablers:

- **V2X Aggregator:** Collects and aggregates BSMs at the RSUs, enabling centralized threat analysis.
- **Reinforcement Learning (RL)-Based DDoS Detector:** Utilizes spatial and temporal patterns in vehicular traffic to identify anomalous behavior. This learning-based approach adapts to evolving attack vectors and minimizes FPs.
- **Filtering Assets:** Implemented at the RSU level, these components filter traffic based on policy thresholds such as BSM inter-arrival times, effectively discarding high-frequency malicious transmissions.
- **5G Core and RAN Security Agents:** These agents perform real-time reconfiguration of radio and core network functions. For instance, they can isolate malicious vehicles by redirecting them to uplink-disabled cells.

Each enabler is associated with a dynamic trust score, guiding the orchestration process during policy enforcement.

3.3 Use Case and Evaluation Setup

To demonstrate the feasibility of the approach, a multi-domain V2X testbed was established across two institutions (CTTC and UMU labs). The testbed deployed an E2E security slice protecting V2X services hosted at the edge level. The framework autonomously deployed, monitored and adapted configurations based on SSLA policies. To generate the set of multiple DDoS attacks, the VeReMi dataset [9] was used in the evaluation phase. The VeReMi dataset comprises the following DDoS attack variants:

1. **DDoS attack:** Malicious vehicles transmit BSMs at a higher frequency than the acceptable limit set by the standard specifications. The frequency threshold considered in this work is 4Hz, which is equivalent to an inter-arrival BSM time threshold of 250ms.

2. **DDoS Random attack:** In this attack, malicious vehicles set all BSM fields to random values and perform a typical DDoS attack.

3. **DDoS Disruptive attack:** The malicious vehicles may re-transmit previously transmitted BSMs by other legitimate vehicles, with the intention of disrupting genuine information from being propagated.

4. **DDoS Random Sybil attack:** The malicious vehicles change pseudonym identities on every transmitted BSM while performing the DDoS random attack.

5. **DDoS Disruptive Sybil attack:** In this attack, the malicious vehicles change pseudonyms on every re-transmission of previously received BSMs while performing the DDoS disruptive attack.

Our selected performance indicators are defined as follows:

- Accuracy: The ratio of all correct predictions (i.e., TPs and TNs) to the total number of considered input samples.

- Precision: The ratio of TPs to the number of TPs plus the number of FPs.

- Recall: The ratio of TPs to the number of TPs plus the number of FNs.

- F2-score: The weighted harmonic mean between precision and recall metrics. The F2-score weights recall higher than precision[7], to account for the higher importance of FNs compared to FPs in safety-threatening V2X scenarios.

- False Positive Rate (FPR): The rate at which a legitimate vehicular behavior is identified as malicious.

- False Negative Rate (FNR): The rate at which a malicious vehicular behavior is identified as legitimate.

- Mean Time to Detect (MTTD): The average time elapsed between the time the DDoS attack takes place and its discovery by the V2X DDoS detector.

- Mean Time to Resolve (MTTR): The average time elapsed between the time the DDoS attack is detected and the enforcement of the mitigation (filtering) policy by the security orchestrator.

[7] In contrast, the traditional F1-score weights equally both recall and precision.

3.4 Performance Results

3.4.1 Secured V2X Slice Deployment Performance

The initial deployment time is an indicator of how long the framework takes from the time the CTTC SMD receives the MSPL-OP until the sub-slice is deployed and configured, and thus the service is ready to be used with all requested capabilities. The distribution of the time consumed by the different processes with respect to the total time can be seen in Figure 10 (left), where the total time has an average of 97.42s, the deployment represents a 59% of the time with 57.73s, the configuration of the V2X DDoS detector 38.99% with 37.99s and the orchestration 1.75% with 1.84s in average. The orchestration time mainly joins the time spent on the translation (0.17s avg.) and creation of the enforcement plan (0.97s avg.) in addition to the load of the different managers and structures. The E2E time is omitted since the other domains do not contain specific V2X enablers. Indeed, the time spent by the E2E SMD is less than 1.8s on average for constructing all the required MSPL-OPs.

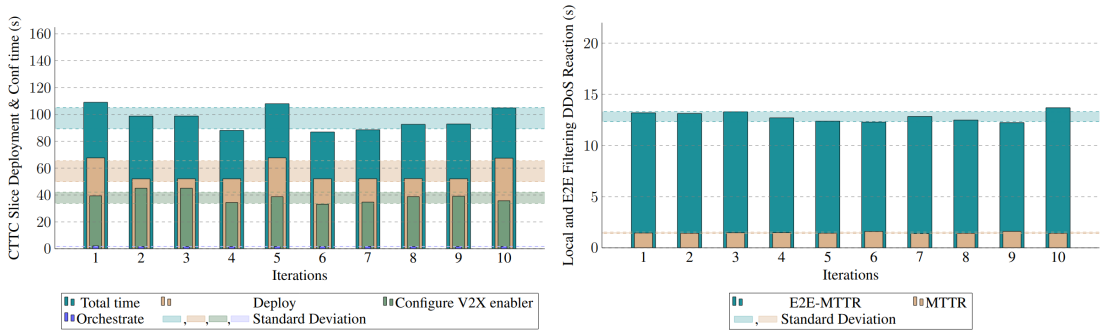

Figure 10: (left) Proactive ZSM-based E2E B5G security slice deployment; (right) Reactive ZSM-aligned trust-based multi-domain mitigation.

Attack type	Accuracy (%)	Precision (%)	Recall (%)	F2 (%)	FPR (%)	FNR (%)	MTTD (ms)	MTTR (s)
DDoS	98.90	99.41	98.72	98.86	1.16	1.40	4.2	1.48
DDoS Random	99.78	99.88	99.66	99.70	0.42	0.45		
DDoS Disruptive	97.96	99.01	96.96	97.36	1.23	3.17		
DDoS Random Sybil	94.76	94.68	94.25	94.34	5.03	5.90		
DDoS Disruptive Sybil	92.94	92.98	92.14	92.31	6.57	8.02		

Table 4: Detection performance per DDoS attack variant.

3.4.2 Detection Performance

Table 4 demonstrates the detection performance per DDoS variant. Results show that RL-based detection is performed effectively, with an F2-score superior to 92.5% for all five DDoS attack types. Notably, the achieved precision and recall values demonstrate the ability of our detector to accurately differentiate DDoS attack messages from genuine behavior. It can also be observed that when DDoS attacks are launched in Sybil mode, detection performance

registers a decline with increased FPR and FNR, albeit not at prohibitive levels. The reported MTTD on average across all DDoS attacks, is in the order of 4ms, which is effective for many road safety applications, as BSMs usually broadcast with frequency of 1-10Hz [37]. Such low detection latency levels corroborate the real-time capabilities of our V2X DDoS detector.

3.4.3 Local and E2E Reaction Performance

The E2E reaction performance refers to the time elapsed since the alert of the DDoS attack is triggered until the framework has mitigated the attack, both at the local domain by the V2X filtering asset, and at the E2E level by banning the vehicles from the cells. As shown in Table 4 and Figure 10 (right), the local mitigation time (MTTR) has an average of 1.48s, which includes the mitigation security policy creation and the enforcement via reconfiguration of the V2X aggregator. Such value is considered acceptable for mitigating the detrimental effects on road users and avoiding the propagation of safety-threatening incorrect information by malicious vehicles. The E2E reaction time (E2E MTTR) has an average value of 12.82s which includes (i) the escalation of the alert to the E2E SMD, (ii) the E2E orchestration (creation of the MSPL-OP for the UMU domain), (iii) the extraction of the required information at the UMU domain for the malicious vehicle from the 5G core, and (iv) the 5G RAN reconfiguration enforcement to ban the malicious vehicle. Both reactions are executed in parallel, thus the total time is equal to the E2E MTTR.

3.5 Summary

This study demonstrates the feasibility of a ZSM-based security framework to manage the complex V2X characteristics in B5G networks. Our approach reveals how the benefits of policy flexibility are leveraged to deploy slices for V2X services with security requirements and an E2E perspective. Such autonomous management of V2X services, driven by a policy-based closed-loop, allows adaptation to the complex and highly dynamic V2X security landscape. Our proposed framework ensures automated reaction to security attacks, not only in the domain where the attacks are launched but also in other domains that could be affected or used to mitigate the attack more efficiently. We assessed the detection capabilities of our framework in the presence of different DDoS attack variants launched by multiple vehicles in a multi-domain V2X scenario. Performance evaluation demonstrated that DDoS attacks can be effectively detected and contained with relatively low latency levels.

4 Misbehavior Detection Using a Hybrid DL Framework

4.1 Introduction

While the DRL-based approach described in Section 2 is tailored toward decentralized V2X environments, we hereby describe a centralized misbehavior detection approach that leverages a hybrid DL framework. In the first stage, unsupervised learning techniques adapt to the dynamic nature of V2X traffic patterns, overcoming the scarcity of labeled data for attacks. This stage efficiently identifies potential anomalies. Building upon this foundation, the second stage employs supervised learning to refine the classification and achieve high accuracy in real time.

4.2 Methodology

By using this two-staged approach we can take advantage of the unsupervised anomaly detection and supervised learning to achieve an efficient and adaptable misbehavior detection system.

4.2.1 Stage 1: Unsupervised Pre-training (Anomaly Discovery)

The first stage of the framework consists of the following steps:

1. Data Preprocessing: Raw V2X data is transformed to extract relevant information. Specifically, it focuses on BSMs and converts them into sequential data.
2. Unsupervised Anomaly Detection: A deep learning model, trained without labeled data, analyzes the BSM sequences. It attempts to reconstruct these sequences and calculates a reconstruction error. A high error indicates a potential anomaly.
3. Thresholding: An algorithm sets a threshold based on the reconstruction errors. This threshold helps distinguish normal BSM sequences from anomalous ones.
4. Label Generation: Using the established threshold, the model classifies each sequence as either normal or anomalous. This creates a labeled dataset for the next stage.

4.2.2 Stage 2: Supervised Learning for Classification

The second stage of the framework consists of the following steps:

1. Supervised Model Training: The labeled dataset generated in stage 1 is used to train a supervised DL model. This model learns the relationship between BSM features and the assigned labels (normal or anomalous).
2. Real-Time Anomaly Classification: During deployment, the trained model receives incoming BSM sequences as input. It analyzes the features within these sequences and classifies them as normal traffic or potential attacks in real time.

This approach leverages unsupervised learning to discover potential anomalies in the initial stage. The subsequent supervised learning stage refines the classification using the generated labels, resulting in a more robust and accurate anomaly detection system.

4.3 Data Preprocessing

4.3.1 Dataset Selection

This evaluation utilized the publicly available VeReMi Extension dataset [9]. Designed as a reference for V2X network misbehavior detection, it was generated through the open-source simulation tool VEINS. While simulated, the scenario reflects real traffic data from the Luxembourg SUMO Traffic (LuST) scenario. The dataset encompasses 19 attack types and includes features like send time, sender IDs (real and pseudo), message ID, and positional information (position, speed, acceleration, heading).

4.3.2 Feature Selection and Windowing

We employed specific features for analysis: send time, X and Y coordinates from position and speed vectors, and sender pseudo ID. Due to the time-series nature of the data, it's segmented into windows. A window size of 20 and a step size of 10 were chosen, aligning with previous works using the VeReMi Extension dataset.

4.3.3 Normalization

Before feeding the data into the models, normalization is performed using a standard scaler. The following equation represents this normalization process:

$$z = (x - u)/s, \tag{20}$$

where z is the normalized data, x is the original data, u represents the mean and s the standard deviation.

4.4 First stage

This stage leverages three different Deep Neural Network (DNN) architectures for unsupervised anomaly detection:

- Model 1 (M1): LSTM Autoencoder: This model employs two encoder and two decoder layers, each containing 1024 and 512 LSTM units, respectively.
- Model 2 (M2): CNN-LSTM: This architecture combines convolutional and recurrent layers. It consists of two convolutional layers with 1024 and 512 units followed by four stacked LSTM layers, each with 512 units.
- Model 3 (M3): CNN-BiLSTM: This model is similar to M2 but utilizes Bidirectional LSTMs (BiLSTM) in the stacked layers. It has two convolutional layers (1024 and 512 units) followed by four stacked BiLSTM layers.

4.4.1 Input

Each model receives sequences of 20 messages with six features per message (X and Y coordinates from position and speed vectors, send time, and sender pseudo ID) as input. The

objective is to identify anomalies by reconstructing the input sequences and analyzing the reconstruction error.

4.4.2 Model Training and Reconstruction Error

During training, each model is exposed only to normal sequences. This allows it to learn the typical patterns and values associated with genuine V2X communication. When presented with a new sequence, the model attempts to reconstruct it. Ideally, the reconstructed sequence should closely resemble the original sequence, resulting in a low reconstruction error (calculated using the mean squared error).

4.4.3 Anomaly Threshold

High reconstruction errors indicate a potential anomaly. A separate thresholding algorithm utilizes these errors to distinguish normal sequences from anomalous ones. This thresholding step plays a crucial role in isolating potential attacks or malfunctions within the V2X network. Figure 11 summarizes the steps of the first stage.

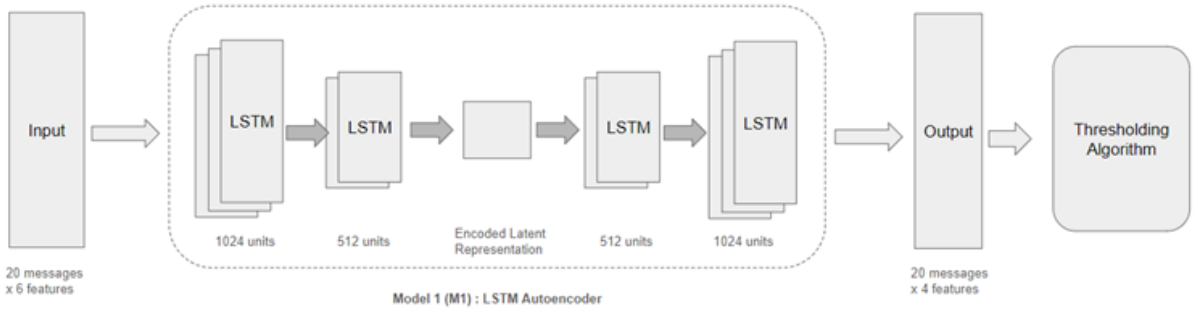

Figure 11: First stage

4.4.4 Thresholding Algorithm

Within the first stage, the thresholding algorithm plays a critical role in classifying sequences as anomalous or genuine. It operates on the reconstruction errors generated by the DNN models. The threshold selection for optimal F1-Score is as follows:

- Threshold Evaluation: The algorithm iterates through a predefined list of potential thresholds. At each step, it calculates the F1-Score of the resulting dataset. The F1-Score is chosen because it balances precision (correctly identified anomalies) and recall (reduced false positives) – crucial factors in real-time V2X applications.

- Performance Analysis: The algorithm analyzes performance metrics (potentially including precision, recall, and F1-Score) across different thresholds. The primary objective is to identify the threshold that maximizes the F1-Score.

- Visualization: As depicted in Figure 12, a visual report can be generated to illustrate the variation of performance metrics with respect to threshold values. This visualization aids in understanding the impact of threshold selection on anomaly detection accuracy.

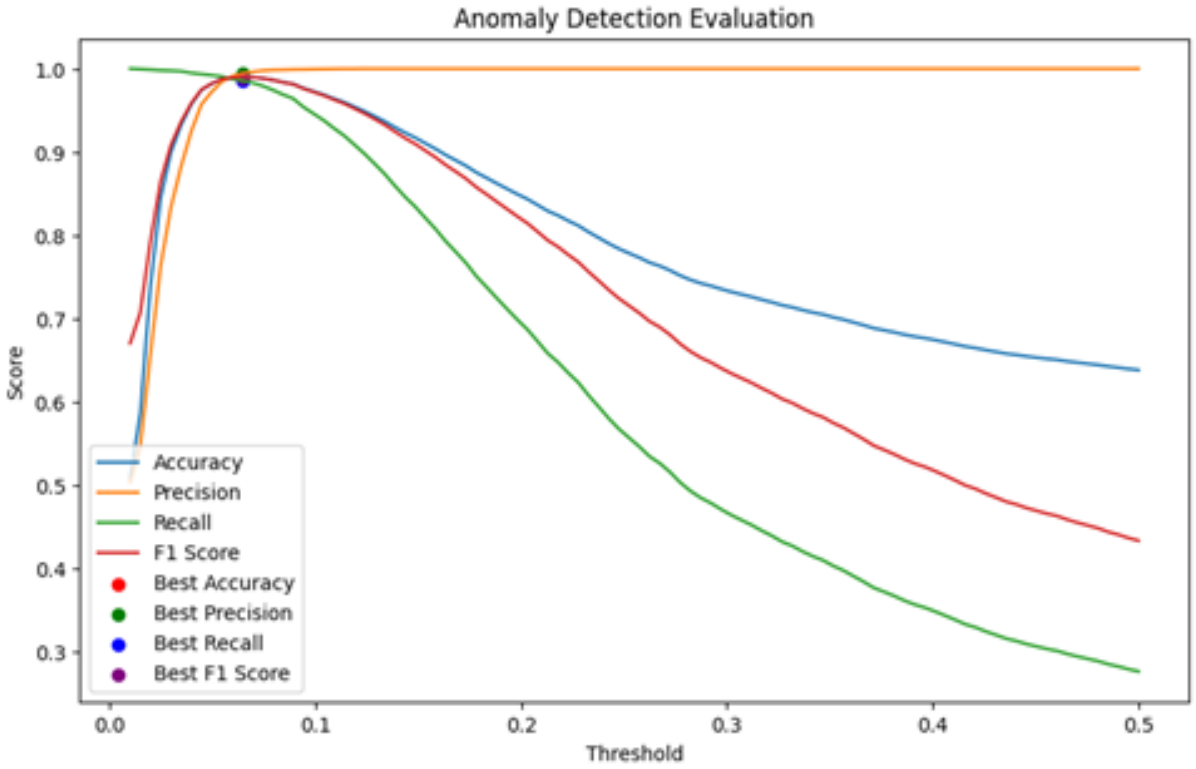

Figure 12: Report Visualization from thresholding algorithm

The chosen threshold is then used to classify each sequence as anomalous (reconstruction error exceeding the threshold) or genuine (error below the threshold). This process creates a labeled dataset that serves as input for the supervised learning stage in the next phase. This approach eliminates the need for extensive manual labeling, which is impractical in real-time V2X scenarios.

4.5 Second stage

The second stage focuses on supervised learning for real-time misbehavior detection during deployment. This section details the configurations of the two DNN models employed:

- Model 1 (S1): CNN: This model leverages Convolutional Neural Networks (CNNs) for classification. It comprises two 2D convolutional layers with 256 and 128 units, respectively. These are followed by two dense layers with 100 and 10 units. The final output layer is a dense layer with a single unit activated by the sigmoid function, suitable for binary classification (attack vs. genuine).

- Model 2 (S2): Stacked LSTM: Inspired by findings in related work, this model utilizes a stacked Long Short-Term Memory (LSTM) architecture. It consists of four LSTM

layers, each containing 256 units. Similar to S1, the final layer is a dense layer with a single unit and sigmoid activation for binary classification.

4.6 Simulation Results

This section presents the obtained results for all the models along with details about the training, testing, and validation data splits.

4.6.1 Training and Testing Setup

- Development Environment: TensorFlow was used for model development.
- Training and Testing Platform: Google Colaboratory was employed for training and testing the models.

Data Splits:

- Unsupervised Learning Dataset:
  - Training set: 66,500 sequences (3,500 genuine sequences from each attack scenario)
  - Testing set: 15,200 sequences (7,600 attack sequences and 7,600 genuine sequences)
  - Rationale: The training set exposes the models to all genuine behaviors by including genuine sequences from each attack scenario.
- Supervised Learning Dataset:
  - Training set: 15,200 sequences (generated by the unsupervised stage)
  - Testing set: 30% of the total dataset (4,560 sequences)
  - Validation set: 10% of the total dataset (1,520 sequences)

Training Details:

- Unsupervised Learning:
  - Optimizer: Adam
  - Loss function: Mean Absolute Error (MAE)
  - Batch size: 20
  - Epochs: 100 (each model)
- Supervised Learning
  - Optimizer: Adam
  - Loss function: Binary Crossentropy
  - Early Stopping: Implemented to prevent overfitting

4.6.2 First Stage Evaluation

Table 5 presents the obtained results for all the models in the first stage.

Model	Class	Precision	Recall	F1-Score	Accuracy
M1	Attack	1.00	0.98	0.99	0.9895
	Genuine	0.98	1.00	0.99	
M2	Attack	0.98	0.96	0.97	0.9719
	Genuine	0.96	0.98	0.97	
M3	Attack	0.99	0.98	0.98	0.9820
	Genuine	0.98	0.99	0.98	

Table 5: First Stage Results

4.6.3 Hybrid Approach Evaluation

Table 6 presents the obtained results for the proposed hybrid approach.

	M1	**S1**	**S2**	**Optimization**
Precision	**0.9964**	0.9865	0.9830	-0.99%
Recall	0.9825	**0.9956**	0.9944	+1.33%
F1-Score	0.9894	**0.9910**	0.9887	+0.17%
Accuracy	0.9895	**0.9911**	0.9887	+0.16%
Parameters	15 Million	**1.3 Million**	2 Million	+91.33%
Prediction Time	0.1666 seconds	**0.0661 seconds**	0.0759 seconds	+60.24%

Table 6: Hybrid Approach Results

References

[1] R. Sedar, C. Kalalas, F. Vázquez-Gallego, L. Alonso, and J. Alonso-Zarate, "A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 325–391, 2023.

[2] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 779–811, 2019.

[3] A. Boualouache and T. Engel, "A Survey on Machine Learning-Based Misbehavior Detection Systems for 5G and Beyond Vehicular Networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1128–1172, 2023.

[4] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, "Poisoning and Evasion Attacks Against Deep Learning Algorithms in Autonomous Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4439–4449, 2020.

[5] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing Vehicle-to-Everything (V2X) Communication Platforms," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[6] A. Talpur and M. Gurusamy, "Adversarial Attacks Against Deep Reinforcement Learning Framework in Internet of Vehicles," in *IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.

[7] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *CoRR*, vol. abs/1810.00069, 2018.

[8] R. Sedar, C. Kalalas, P. Dini, F. Vázquez-Gallego, J. Alonso-Zarate, and L. Alonso, "Knowledge transfer for collaborative misbehavior detection in untrusted vehicular environments," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 1, pp. 425–440, 2025.

[9] J. Kamel, M. Wolf, R. W. van der Hei, A. Kaiser, P. Urien, and F. Kargl, "VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," in *ICC 2020 - IEEE International Conference on Communications*, 2020, pp. 1–6.

[10] R. Sedar, C. Kalalas, F. Vázquez-Gallego, and J. Alonso-Zarate, "Reinforcement Learning Based Misbehavior Detection in Vehicular Networks," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 3550–3555.

[11] R. Sedar, C. Kalalas, P. Dini, J. Alonso–Zarate, and F. Vázquez–Gallego, "Misbehavior Detection in Vehicular Networks: An Ensemble Learning Approach," in *GLOBECOM 2022 - IEEE Global Communications Conference*, 2022, pp. 1–6.

[12] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2017, pp. 262–275.

[13] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2015.

[15] M. Yuan, M. Pun, Y. Chen, D. Wang, and H. Li, "Multimodal reward shaping for efficient exploration in reinforcement learning," 2021.

[16] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer Learning in Deep Reinforcement Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 344–13 362, 2023.

[17] Y. Tao, S. Genc, J. Chung, T. Sun, and S. Mallya, "Repaint: Knowledge transfer in deep reinforcement learning," in *International conference on machine learning*. PMLR, 2021, pp. 10 141–10 152.

[18] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning," in *International Conference on Learning Representations*, 2022.

[19] K. Lin, S. Wang, and J. Zhou, "Collaborative deep reinforcement learning," *arXiv preprint arXiv:1702.05796*, 2017.

[20] J. García, Á. Visús, and F. Fernández, "A taxonomy for similarity metrics between markov decision processes," *Machine Learning*, vol. 111, no. 11, pp. 4217–4247, 2022.

[21] F. Lotfi, O. Semiari, and W. Saad, "Semantic-aware collaborative deep reinforcement learning over wireless cellular networks," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 5256–5261.

[22] J. Guo, X. Li, Z. Liu, J. Ma, C. Yang, J. Zhang, and D. Wu, "TROVE: A Context-Awareness Trust Model for VANETs Using Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6647–6662, 2020.

[23] S. Gyawali, Y. Qian, and R. Q. Hu, "Deep Reinforcement Learning Based Dynamic Reputation Policy in 5G Based Vehicular Communication Networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6136–6146, 2021.

[24] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 3878–3887.

[25] F. S. He, Y. Liu, A. G. Schwing, and J. Peng, "Learning to play in a day: Faster deep reinforcement learning by optimality tightening," in *International Conference on Learning Representations*, 2017.

[26] M. C. Lucas-Estan, B. Coll-Perales, C.-H. Wang, T. Shimizu, S. Avedisov, T. Higuchi, B. Cheng, A. Yamamuro, J. Gozalvez, M. Sepulcre, and O. Altintas, "On the Scalability of the 5G RAN to Support Advanced V2X Services," in *2020 IEEE Vehicular Networking Conference (VNC)*, Dec. 2020, pp. 1–4.

[27] A. Ghosal and M. Conti, "Security issues and challenges in V2X: A Survey," *Computer Networks*, vol. 169, p. 107093, Jan. 2020.

[28] F. Sakiz and S. Sen, "A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV," *Ad Hoc Networks*, vol. 61, pp. 33–50, Mar. 2017.

[29] N. Salhab, R. Langar, and R. Rahim, "5G network slices resource orchestration using Machine Learning techniques," *Computer Networks*, vol. 188, p. 107829, Apr. 2021.

[30] R. Vilalta, P. Alemany, R. Sedar, C. Kalalas, R. Casellas, R. Martínez, F. Vázquez-Gallego, J. Ortiz, A. Skarmeta, J. Alonso-Zarate, and R. Muñoz, "Applying Security Service Level Agreements in V2X Network Slices," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2020, pp. 114–115.

[31] S. Zhang and D. Zhu, "Towards artificial intelligence enabled 6G: State of the art, challenges, and opportunities," *Computer Networks*, vol. 183, p. 107556, Dec. 2020.

[32] N. F. Saraiva de Sousa, D. A. Lachos Perez, R. V. Rosa, M. A. Santos, and C. Esteve Rothenberg, "Network Service Orchestration: A survey," *Computer Communications*, vol. 142-143, pp. 69–94, Jun. 2019.

[33] C. Benzaid and T. Taleb, "AI-driven zero touch network and service management in 5g and beyond: Challenges and research directions," vol. 34, no. 2, pp. 186–194. [Online]. Available: https://ieeexplore.ieee.org/document/8994961/

[34] M. M. Sajjad, C. J. Bernardos, D. Jayalath, and Y.-C. Tian, "Inter-slice mobility management in 5g: Motivations, standard principles, challenges, and research directions," *IEEE Communications Standards Magazine*, vol. 6, no. 1, pp. 93–100, 2022.

[35] R. Asensio-Garriga, P. Alemany, A. M. Zarca, R. Sedar, C. Kalalas, J. Ortiz, R. Vilalta, R. Muñoz, and A. Skarmeta, "ZSM-Based E2E Security Slice Management for DDoS Attack Protection in MEC-Enabled V2X Environments," *IEEE Open Journal of Vehicular Technology*, vol. 5, pp. 485–495, 2024.

[36] R. Sedar, C. Kalalas, J. Alonso-Zarate, and F. Vázquez-Gallego, "Multi-domain Denial-of-Service Attacks in Internet-of-Vehicles: Vulnerability Insights and Detection Performance," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 438–443.

[37] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions," Tech. Rep. Standard ETSI TR 102 638 V1.1.1, Jun. 2009.