Adaptive Attack Mitigation for IoV Flood Attacks

Erol Gelenbe, *Fellow, IEEE,* and Mohammed Nasereddin

***Abstract*—Gateway Servers for the Internet of Vehicles (IoV) must meet stringent Security and Quality of Service (QoS) requirements, including cyberattack protection, low delays and minimal packet loss, to offer secure real-time data exchange for human and vehicle safety and efficient road traffic management. Therefore, it is vital to protect these systems from cyberattacks with adequate Attack Detection (AD) and Mitigation mechanisms. Such attacks often include packet Floods that impair the QoS of the networks and Gateways and even impede the Gateways' capability to carry out AD. Thus, this paper first evaluates these effects using system measurements during Flood attacks. It then demonstrates how a Smart Quasi-Deterministic Policy Forwarder (SQF) at the entrance of the Gateway can regulate the incoming traffic to ensure that the Gateway supports the AD to operate promptly during an attack. Since Flood attacks create substantial packet backlogs, we propose a novel Adaptive Attack Mitigation (AAM) system that is activated after an attack is detected to dynamically sample the incoming packet stream, determine whether the attack is continuing, and also drop batches of packets at the input to reduce the effects of the attack. The AAM is designed to minimize a cost function that includes the sampling overhead and the cost of lost benign packets. We show experimentally that the Optimum AAM approach is effective in mitigating attacks and present theoretical and experimental results that validate the proposed approach.**

***Index Terms*—Cyberattack Detection and Mitigation, Internet of Vehicles, Flood Attacks, Quasi-Deterministic Transmission Policy, Adaptive Attack Mitigation**

I. Introduction

Smart vehicles rely on many onboard and road-side IoT devices, and today's 30 Billion or more IoT devices [1] are already subjected to numerous cyberattacks [2], [3] which disable systems with huge packet floods [4]. For instance, an attack in 2017 took down $180,000$ servers with an overall 2.54 Terabits per second of traffic [5]. Thus, it is imperative to develop the understanding and technology that can keep both the IoV and the IoT as safe as possible.

Unlike the general IoT which mainly involves stationary, low-power [6] or occasionally mobile devices, the IoV operates mostly with numerous continuously moving nodes, i.e. vehicles, that generate, process, and transmit data in real-time, together with road-side monitoring and relaying systems whose complexity may cause the random misdirection and loss of data [7]. The safety-critical nature of the IoV, which directly impacts road safety and traffic efficiency, demands stringent requirements on data timeliness, reliability, and very low latency [8], [9], making it particularly vulnerable to Flood attacks, which overloads the network's gateway servers that act as control hubs which manage the flow of data between vehicles and the broader network infrastructure, and severely compromise QoS.

Smart vehicles may number two billion by 2035 [10], so supply chains and manufacturing networks will also rely in the future on smart transport [11]. Since smart vehicles depend on timely and accurate data collection and forwarding, cyberattacks against smart vehicles can cause network delays, data loss and inaccuracy, resulting in road traffic gridlock, driver and passenger frustration, delayed delivery of goods and services, and wrong vehicle traffic control decisions [12] that can lead to road congestion and impair safety [13].

Therefore, the present paper proposes an Attack Mitigation approach that operates together with an Attack Detector (AD), that acts at the input of an IoV or IoT Gateway to detect attacks rapidly in real time to optimally mitigate their effects. The AD is based on previous work on highly accurate Machine Learning (ML) attack detection methods [14]–[16] which determine whether the incoming traffic deviates from the expected normal behaviour. The novel Adaptive Attack Mitigation (AAM) scheme that we propose, rapidly drops incoming attack packets when the AD produces an alarm and actively samples the incoming packet stream in a manner that minimizes the Gateway overhead and the cost of mistakingly dropping benign packets.

A. Related Work on the Security of the IoV and IoT

The literature on cybersecurity for the IoV and the IoT, encompasses a range of studies addressing attack and defense methodologies, proposed solutions, and related challenges. Several security architectures, lightweight protocols, and frameworks for autonomous vehicle communication are discussed in [17], emphasizing the integration of advanced security solutions. In [18], attack and defense studies are categorized to shed light on methods and to identify security challenges for future exploration. In the realm of autonomous driving, the integration of artificial intelligence (AI) into networks and network control algorithms, raises additional cybersecurity concerns that must be considered [19], [20].

Cyberattack detection algorithms may be trained offline with calibrated real or synthetic datasets, or online with real data. Most studies then evaluate the accuracy of the resulting

Manuscript received XX, 2024; revised YY, 2024.

This research was supported in part by the Horizon Europe Program DOSS Project under Grant Agreement 101120270.

Prof. Erol Gelenbe is with the Institute of Theoretical & Applied Informatics, Polish Academy of Science, 44-100 Gliwice, PL, and is also an Honorary Researcher at CNRS I3S, Université Côte d'Azur, 06100 Nice, FR, and a Visiting Professor in the Department of Engineering, King's College London, Strand WC2R 2LS, UK. ORCID: 0000-0001-9688-2201, email: gelenbe.erol@gmail.com

Mohammed Nasereddin is with the Institute of Theoretical & Applied Informatics, Polish Academy of Science, 44-100 Gliwice, PL. ORCID: 0000-0002-3740-9518, email: mnasereddin@iitis.pl

TABLE I
PERFORMANCE COMPARISON OF ATTACK DETECTORS FOR IOV NETWORKS

CyberAttack Detection for the IoV	**Detection Method**	**Evaluation Strategy**	**Evaluation Metrics: Accuracy || TPR || TNR**
"Secure attack detection framework for hierarchical 6G-enabled internet of vehicles" [28]	Hybrid (Federated Learning (FL) + Stackelberg Game)	Simulation using the UNSW-NB15 dataset	96% || 95% || 97%
"A Transfer Learning and Optimized CNN Based Intrusion Detection System for Internet of Vehicles" [29]	Deep Learning (Transfer Learning + Optimized CNN)	Car-Hacking and CICIDS2017 datasets	99.93% || 99.90% || 99.82%
"An Intrusion Detection System for Connected Vehicles in Smart Cities" [30]	Hybrid (Deep Belief Network (DBN)+ Decision Tree (DT))	Simulation using ns-3 and NSL-KDD datasets with validation through MATLAB simulations	99.43% || 99.04% || 98.47%
"Cybersecurity for Autonomous Vehicles Against Malware Attacks in Smart Cities" [31]	Hybrid (Static + Dynamic Analysis with ML)	Evaluated using a custom dataset of 1000 malware downloaded from *Virusshare.com* and 1000 non-malware applications	96.3% || 95.8% || 97.2%
"Cybersecurity in Automotive: An Intrusion Detection System in Connected Vehicles" [32]	Hybrid (Bayesian Networks + Spatial-Temporal Analysis)	Evaluated using simulated datasets generated through the CARLA simulator for various attack scenarios	98.2% || 97.8% || 98.5%
"MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles" [33]	Hybrid (Tree-based Ensemble Learning + Clustering)	Cross-validation on CAN-intrusion and CICIDS2017 datasets	99.99% (CAN) || 99.88% (CICIDS2017) 99.81% || 99.89%
"Detection and identification of malicious cyber-attacks in connected and automated vehicles' real-time sensors" [34]	Hybrid (Discrete Wavelet Transform + Bayesian DL)	Evaluated on simulated data generated from the Safety Pilot Model Deployment dataset	98.3% || 97.9% || 98.1%
Our System in this Paper	**Hybrid (Random Neural Network + Adaptive Attack Mitigation)**	**Evaluated on a custom testbed with flood attacks implemented using the MHDDoS public repository**	**99.71% || 99.73%|| 98.48%**

trained ADs with real or synthetic datasets, using the following performance scores that are obtained from the total number of true positives (TP), true negatives (TN), false positives (FP) or false alarms, and false negatives (FN), obtained during testing:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$
$$True\ Positive\ Rate\ (TPR) = \frac{TP}{TP + FN},$$
$$True\ Negative\ Rate\ (TNR) = \frac{TN}{TN + FP}.$$

In some cases, other metrics such as the $F1$ and $F2$ scores are also used.

The research on ADs, their design [21] and accuracy [22], [23], often use ML methods that are evaluated off-line [24]–[27] with various datasets, rather than with on-line traffic traces. Such evaluations typically neglect the overload created by attacks on the victim devices or systems, which can paralyze the targeted system with floods of packets, in addition to compromising them with malware.

Some cyberattack test-beds [35] have been constructed for experimentation [36]–[38] in critical applications such as wind farms [39], and Supervisory Control and Data Acquisition (SCADA) systems [40], [41]. Experiments with Flood attacks were presented in [42], and real-time DNS attack data was collected in [43]. Software Defined Networks used for smart network routing, that adapt the packet flows to the needs of smart vehicles, can also be subjected to Denial of Service (DoS) attacks [44]. Attack emulation, without considering the effect of the overload that attacks produce, was investigated for autonomous vehicles in [45].

ML and Deep Learning (DL) take center stage in many AD studies. For instance, a Unified Modeling Language (UML) based framework [46] introduces Decision Tree and Naive Bayes algorithms to classify vulnerabilities. In a bid to fortify vehicle network security, an innovative AI-based solution [47] employs Convolutional Neural Networks (CNN) and Hybrid CNN Long and Short-Term Memory (CNN-LSTM) models for the detection and classification of message attacks. Additionally, the abnormal behavior within vehicle networks has been studied with a classification that uses a generative adversarial network (GAN) in [48].

Security concerns for the IoV have received further attention in [49] with DL to reduce false positives and enhance the resilience of the transportation ecosystem. In [50], [51], data-centric misbehavior detection based on supervised ML and transfer learning has also been proposed. A holistic approach in [28] presents a detection framework tailored for 6G-enabled IoV, with edge node processing, federated learning, and robust security measures.

TABLE II
COMPARISON OF MITIGATION APPROACHES IN IOV NETWORKS

Reference	Mitigation Approach	Brief Description	Evaluation Strategy	Evaluation Results	Limitations
"A Comprehensive Detection and Mitigation Mechanism to Protect SD-IoV Against DDoS Attacks " [57]	Hybrid (Entropy + Flow Rate Analysis)	Utilizes entropy metrics to detect anomalies by analyzing payload lengths and mitigates attacks using adaptive thresholds	Simulated using SUMO, Mininet-WiFi, and Scapy on a SD-IoV environment with different scenarios	Reduced network load and bandwidth congestion by blocking IPs that send a number of packets exceeding the identified threshold for a specified period, before re-evaluating them	May not scale for high-volume traffic, as entropy and flow rate analysis over sliding windows can be computationally intensive, causing real-time delays |&| Adaptive thresholds require careful tuning to handle varying network conditions to avoid false positives
"A Novel DDoS Mitigation Strategy in 5G-Based Vehicular Networks Using Chebyshev Polynomials" [58]	Chebyshev Polynomials-based	Leverages Chebyshev polynomials for lightweight cryptographic authentication to reduce computational overhead in 5G IoV networks during DDoS attacks	Analytical evaluation focused on cryptographic efficiency	Significantly reduced computational cost while enhancing system responsiveness under high traffic loads	A larger message-signature tuple increases communication overhead, potentially causing latency in dense networks |&| It has not been evaluated in real-time vehicular scenarios
"Optimized Feature Selection for DDoS Attack Recognition and Mitigation in SD-VANETs" [59]	Hybrid (Statistical Analysis + ML)	Uses optimized feature selection with LSTM models for effective mitigation, focusing on reducing false positives	Emulated on real datasets with statistical feature analysis	Achieved 94% detection accuracy with optimized mitigation strategies	It requires high computational resources, challenging for low-powered vehicular devices
"DDoS Mitigation Based on Space-Time Flow Regularities in IoV" [60]	Reinforcement Learning (RL)	Uses RL to adaptively adjust mitigation strategies based on space-time flow patterns and disconnects malicious connections through dynamic feature selection	Tested on the Shenzhen taxi-cab dataset using the simulation tools 'ddosflowgen' and 'hping3	Yielded high detection accuracy, but with increased time and memory consumption	It needs high computational demand as it uses big training data, limiting deployment in real-time IoV environments
"RSU-Based Online Intrusion Detection and Mitigation for VANET" [61]	Statistical Anomaly Analysis	Uses roadside units (RSUs) to filter and block malicious traffic in real-time and focuses on stealthy DDoS mitigation using statistical methods to identify abnormal data streams	Evaluated using real traffic datasets in a simulation environment	Effectively reduced the impact of attacks on urban RSUs with low false positive rates and delays	Less effective in areas with low RSU density |&| relying on centralized infrastructure
Our System in this Paper	**Smart Quasi-Deterministic Forwarding policy and Adaptive Attack Mitigation**	**Implements a smart forwarding mechanism to regulate incoming traffic and uses adaptive packet dropping to maintain network devices' performance during flood attacks**	**Evaluated on a custom testbed with different flood attack scenarios implemented using the MHDDoS public repository**	**Significantly minimized the cost function that accounts for both benign packet drops and the overhead associated with testing incoming packets**	**Dropping benign packets can be a challenge during long-time attacks |&| Highly sensitive to parameter tuning in complex IoV environments**

The IoT and IoV can also use different devices from different vendors that may not wish to share their network datasets. For instance, the healthcare IoT may use body sensors, hospital security devices, and patient monitoring, while the IoV may be used for distinct vehicles that convey patients. Thus recent research has also investigated cyberattack detection methods that use transfer learning between distinct IoT devices and distinct vehicles that do not wish to share data, but which need to take advantage of the cybersecurity experience of their peers [52]. Transfer and ensemble learning using CNNs have also been combined to achieve impressive attack detection rates that exceed 99.25% [29]. In [53], where DL is deployed on Multi-access Edge Computing (MEC) servers, demonstrating great efficacy in classifying potential cyberattacks and enhancing real-time security for IoV networks. Expanding the focus to connected vehicles in smart cities, an AD system utilizing spatial and temporal analysis and Bayesian networks is proposed in [30], while a hybrid approach combining static and dynamic analyses is discussed in [31]. In [32], [33], embedded AD systems for the automotive sector are presented. Another example combining Bayesian DL and Discrete Wavelet Transform for proactive detection can also be found in [34].

Since many types of attacks may target the Authentication, Availability, Secrecy, Routing, and Data Authenticity in automobile networks [54], [55], the dynamic nature of the IoV and the presence of varied cyberattacks have also been addressed with transfer learning using cloud-assisted updates [56].

In addition to the previous references, seven state-of-the-art attack detection methods for IoV attacks are discussed in detail and compared with the method that we use in this paper in Table I. Furthermore, five recent mitigation techniques for IoV attacks are also discussed and compared with the method that we have developed in this paper in Table II.

B. Organization of this Paper

The previous Section I-A has reviewed the literature on AD and Mitigation for the IoT and the IoV, and Tables I and II compare recent state-of-the-art results with the work in the present paper.

Our previous work [62] presents experiments that show the impact of Flood Attacks that often accompany cyberattacks on IoV Servers or Gateways, such as C-ITS Road Side Infrastructure Installations for city traffic control and dispatch [13], and IoT Gateways for smart homes, buildings and factories [63].

In Section II, experimental results are shown concerning the system presented in Figure 1, which is comprised of a Gateway Server that supports an AD system and a set of connected devices that may forward benign or attack traffic to the Server. In our experiments, a compromised Sensor sends UDP Flood attacks against the Server which receives network traffic from several Sensors. The measurements that are exhibited in this section, and which are not included in our earlier work [62], demonstrate the impact of the attack on the IoT Server, which creates congestion and **disrupts the Server's operations** by impeding it from carrying out its important role of supporting the AD system that detects the attacks.

We then review the use of a Smart "Quasi-Deterministic Forwarding Transmission Policy (QDTP)" Forwarder (SQF) [64] in Section III. In particular, we evaluate the performance of the architecture shown in Figure 6 that includes the SQF, whose role is to shape the incoming traffic before it enters the Server. Our analysis and experimental results show that by choosing the SQF parameters judiciously, the undesirable effects of the Flood Attack on the Server are eliminated.

However, the length of the input queue to the SQF increases significantly during the attack, and therefore a mitigation algorithm under the control of the AD system is needed.

Therefore, in Section IV, a novel attack mitigation algorithm using optimum AD and packet drop functions named "Optimum AAM" is proposed and its parameters are optimized. Optimum AAM is also extensively validated with experimental results and measurements. Its scalability is discussed in Section IV-C.

Finally, Section V presents our conclusions and suggestions for future work.

II. The Experimental Test-Bed and its Behaviour Under a Flood Attack

The test-bed of Figure 1 includes sensors emulated by Raspberry Pi 4 Model B Rev 1.2 (RPi1 and RPi2) computers. Each RPi is equipped with a $1.5GHz$ ARM Cortex-$A72$ quad-core processor, $2GB$ $LPDDR4-3200$ SDRAM, and runs on Raspbian GNU/Linux 11 (Bullseye), a Debian-based operating system. One RPi is programmed to send attack traffic packets randomly or in a predetermined manner, while the (uncompromised) RPi sends legitimate UDP packets that contain real data about the machines' temperature periodically to the Server. The RPis have a network buffer size of 176 kilobytes (KB) for both incoming and outgoing data.

The Server is an Intel $8-Corei7-8705G$ processor running at $3.1GHz$, with $16GB$ of RAM, a $500GB$ hard drive, and the Linux $5.15.0-60-66-UbuntuSMP$ operating system. It communicates with low-cost Raspberry Pi processors that emulate sensors through a shared Switch via Ethernet. Figure 2 shows that the Server receives packets through the LAN using the UDP protocol and processes them using the Simple Network Management Protocol (SNMP) version $6.2.0-31$-generic. The Server's network interface card (NIC) supports a maximum speed of $1000Mb/s$ in full duplex mode, with a network buffer size of $208KB$ for incoming and outgoing data. Since UDP does not create connections and avoids ACKs, its lightweight nature is useful for communications with simple sensors and actuators [65].

The maximum size of the transmission unit (MTU) for both the Server and the RPi devices is set to $1.5KB/Packet$, which optimizes packet size for efficient transmission. During a test with 1000 successive packets, the measured latency ranged from $0.082ms$ to $0.514ms$, with an average latency of $0.437ms$, that indicates low latency across the network.

The Server also has an attack detection (AD) system detailed in [15]. It is based on the Random Neural Network [66] extended with triggered movement of neuronal action potentials [67], trained with DL [68]. The dataset that is used is MHDDoS [69], including up-to-date real-world DoS attacks and 56 different attack emulators.

As an example, Figure 3 shows measurements from a Flood Attack that lasts for 60 seconds (sec) and overwhelms the capabilities of the 8-Core Server via an incoming flow of $400,000$ packets. The server becomes paralyzed to the point where it is unable to operate the AD system. As a result, a huge packet queue forms at the Server's input, and it can only be progressively removed after a very long 300 minute (min) delay. This also impairs the Server's other normal activities, such as processing benign packets and removing the packets that are part of the attack.

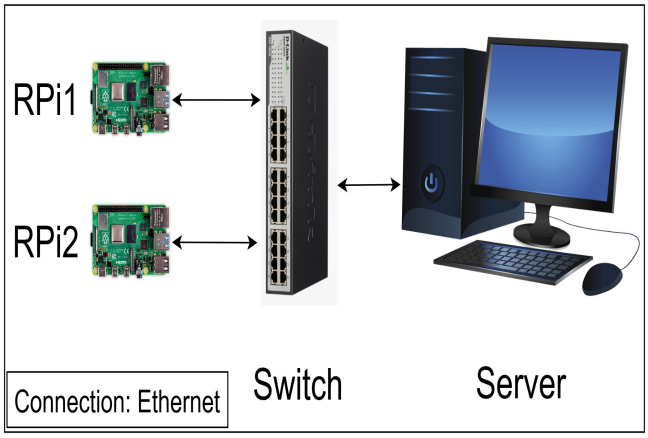

Fig. 1. This figure shows the experimental test bed with several Raspberry Pi machines that emulate various devices, and are connected via Ethernet to the Server. The Raspberry Pis can send both normal and attack traffic to the Server that acts as a Gateway for the IoV

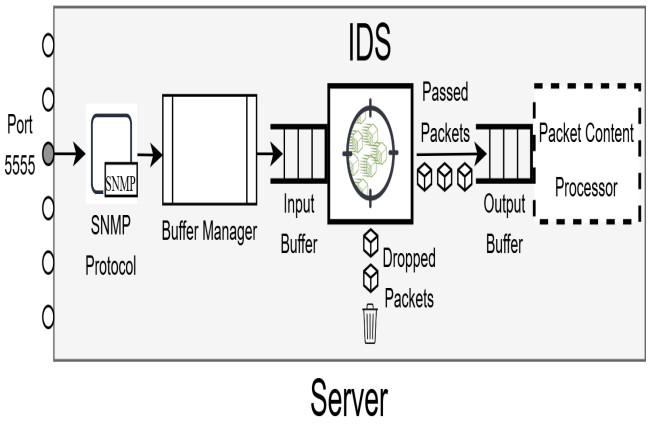

Fig. 2. The software at the Gateway Server includes the manager for the SNMP network, the attack detection system (AD) [15], as well as the software for processing the contents of incoming packets.

A. The Server's Behaviour during Attack Detection

Detailed measurements of the Server's AD processing times per packet both under normal conditions and during a UDP flood attack, are reported in Figure 4 and Figure 5.

Figure 4 (above), indicates that the average AD processing time per packet when there are **no attacks** is 2.98 milliseconds (ms), while (below) we see that when the Server is under a Flood Attack, the average processing time of the AD algorithm rises significantly to 4.82 ms. When the Server is under attack, the AD processing time has very large outliers, as shown both in the histogram of Figure 4 (below), and in Figure 5. These outliers appear to be occasionally 10^3 times larger than the average. As a result, we conclude that the Flood Attack can significantly paralyze and slow down the Server's AD processing rate, as also illustrated in Figure 3.

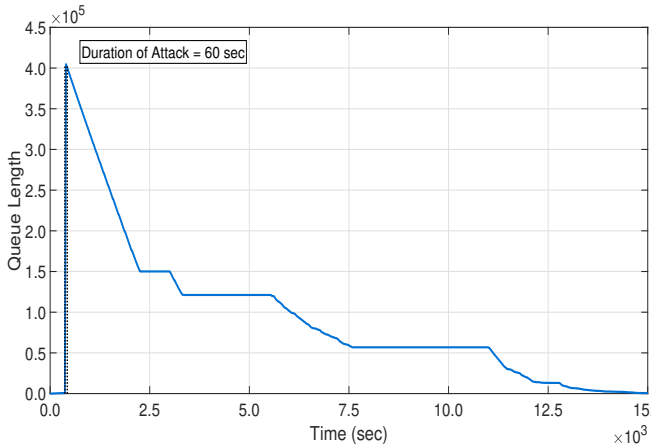

Fig. 3. The queue length shown along the y-axis (number of packets) at Server input prior to the AD, shown as it varies over time (x-axis in second the Server input, prior to processing by the AD module, for a $60-seco$ UDP Flood Attack that was launched by a Raspberry Pi in Figure 1. queue length rises rapidly to $400,000$, and the Server congestion then l far longer than the attack itself, i.e. up to several hours, because of Se paralysis which delays AD processing, as seen in the AD processing ti of Figure 5.

Thus, the **AD algorithm itself is paralyzed** during a U Flood Attack, since the Server is overwhelmed by the SNI protocol which processes the incoming packets.

B. Lindley's Equation when the SQF is not Used

Let us now place ourselves in the context where the SQF module is **not** being used (see Figure 1):

- Let $0 = a_0 \leq a_1 \leq a_2, \ \ldots$, be the successive packet arrival instants at the Server through the Ethernet LAN from any of the Sensors connected to the LAN. We also define the interarrival time $A_{n+1} = a_{n+1} - a_n$.
- Let T_n denote the Server's AD processing time for the n th packet, and assume that the Server processes packe in First Come First Served (FCFS) order.

Then the total waiting time L_{n+1} of the $n+1$-th incomi packet, between the instant a_n and the start of the A processing time of the Server, is given by the well-knov Lindley's equation:

$$L_{n+1} = \max(0, L_n + T_n - A_{n+1}), \ n \geq 0, \ L_0 = 0 \ . \quad ($$

Note that $L_0 = 0$ because the first incoming packet encounte an empty queue in front of the AD. Note also that whenev we have $T_n > A_{n+1}$ then $L_{n+1} > L_n$, i.e. the waiting tir increases.

During a Flood Attack, the values of A_n and T_n will modified, as we see from Figure 3, indicating that packet arrival rates have considerably increased so that the values of A_n are much smaller, while Figure 4 (below) shows that the values of T_n are also larger. However, the form of (1) does not change.

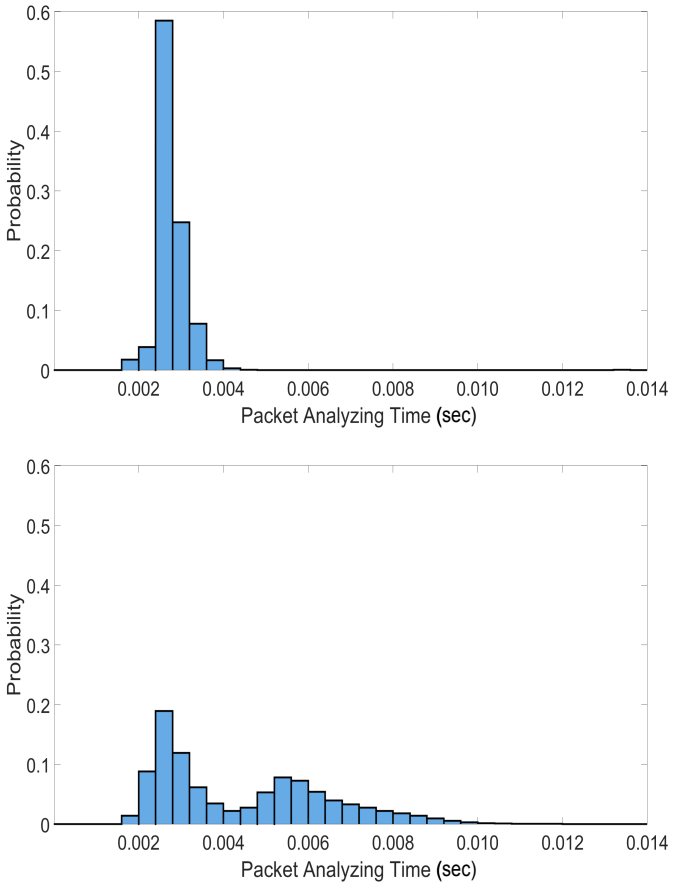

Fig. 4. In the figure that is above, one can observe the histogram of AD processing time per packet, as measured without an attack. It shows the average processing time of 2.98 ms (milliseconds), with a variance of $0.0055\ ms^2$. In the figure given below, an attack occurs and the AD packet processing time increases to the average value of 4.82 ms with $0.51\ ms^2$.

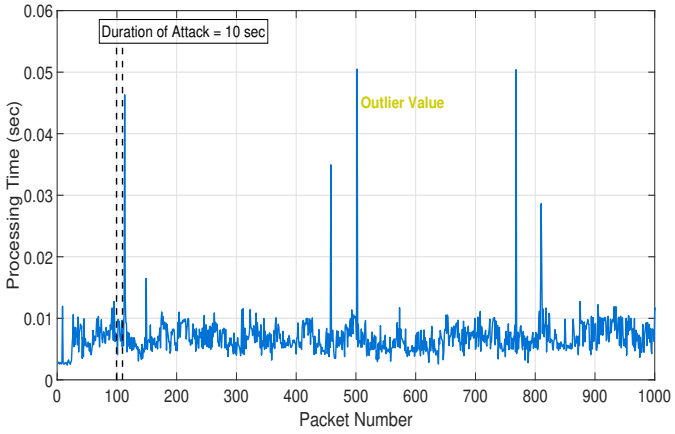

Fig. 5. During a UDP Flood Attack, we show successive measurements for the AD packet processing time per packet when the QDTP Forwarder SQF is not used. The large outliers in processing time that are observed in Figure 4 (below) also confirm the measurements that are shown in this figure.

III. Effect of the Smart QDTP Forwarder (SQF)

In Figure 6, we present our proposed modified architecture where the Server, whose role is to process incoming IoV packets – including operating the AD module in order to detect attacks – is "protected" by a Smart QDTP Forwarder (SQF) placed between the Ethernet output and the Server's

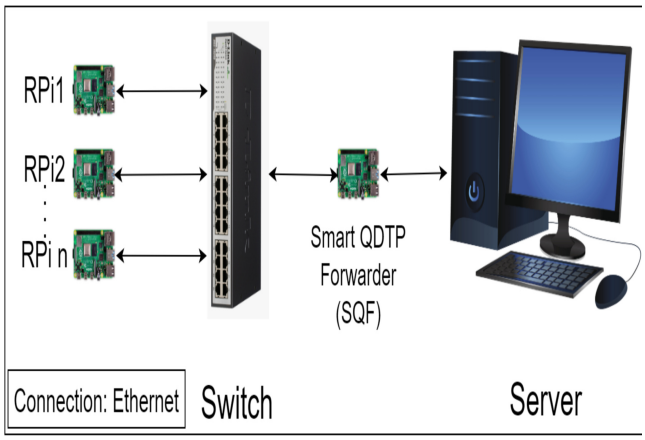

Fig. 6. The figure shows the modified system architecture where a Sma QDTP Forwarder (SQF) is placed before the Server, and acts as a traf shaping interface between the Ethernet LAN and the Server. The effect of t SQF is to eliminate the paralyzing effect of the packet flood at the Serv buffering packets within the SQF, and forwarding the packets, so that A processing and other work are conducted in a timely fashion.

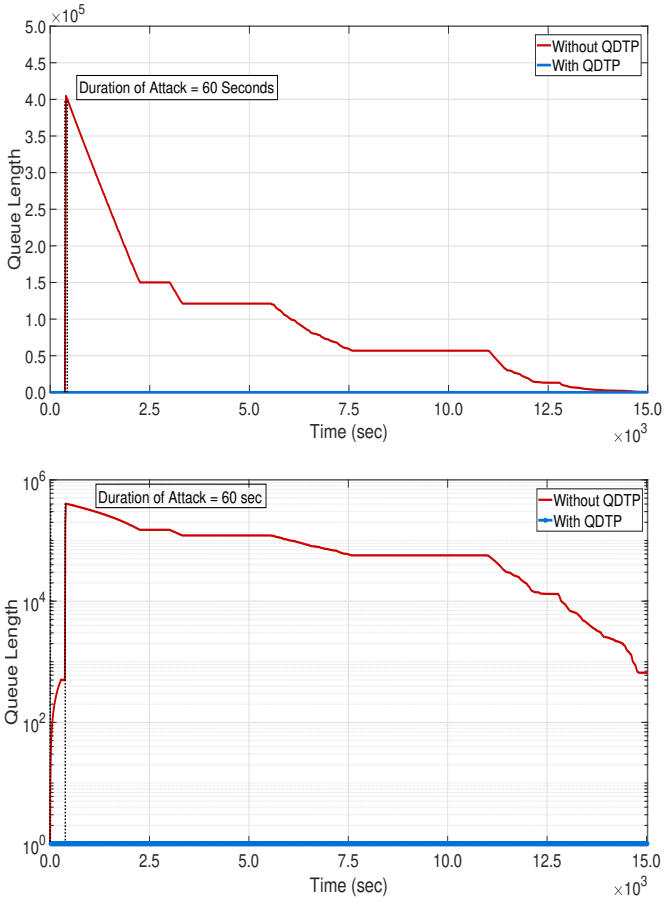

Fig. 7. The Server queue length shown in the figure above, when a 60-second UDP Flood Attack occurs, peaks at $400,000$ packets when the SQF is not used, and drops very slowly during some $15,000$ secs. In the figure that is below, we observe the queue length in logarithmic scale when the SQF is used with $D = 3\ ms$, versus the case shown in Red when the SQF is not used when the Flood Attack lasts 60 secs. Since D is close to the average of T_n (which is ≈ 2.98 ms when an attack does not occur, see Figure 4), small fluctuations of T_n can cause the queue to build up moderately (in Blue).

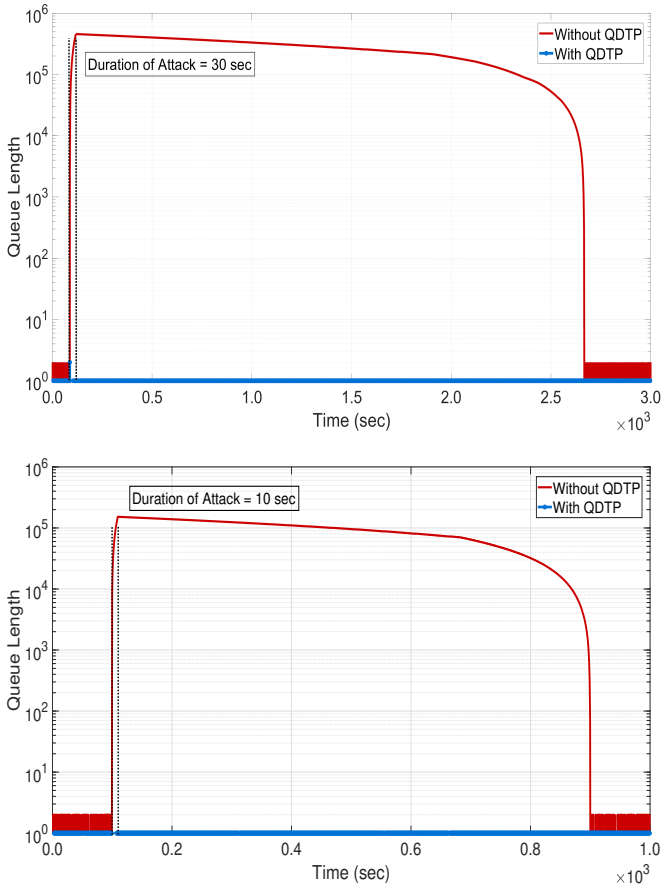

Fig. 8. The Server queue length is measured, and represented logarithmically, for a UDP Flood Attack that targets the Server and lasts 30 sec (above) or 10 sec (below). In Red, we show the queue lengths when the SQF is not used. The effect of using the SQF is shown in the Blue curves, demonstrating the effect of the SQF in reducing queue length during 30 and 10 sec attacks, with $D = 3\ ms$.

input port. The SQF shapes the incoming traffic to the Server with a *Quasi-Deterministic Transmission Policy (QDTP)* [64] that delays some of the packets it receives, by forwarding them to the Server at time $t_n \geq a_n$, where a_n is the n-th packet's arrival instant to the SQF.

We assume that the physical transmission time from the SQF to the Server, and the network protocol service time inside the Server, are negligible with respect to the Server's AD processing duration T_n. This is consistent with our measurements with 1000 packets, that revealed a latency between $0.082ms$ and $0.514ms$, with an average value of $0.437ms$, which is less than 15% of the value of T_n (which is around $3\ ms$).

Thus, when the $n-th$ packet is transmitted by the SQF, it is assumed that it instantly arrives at the queue at the Server's input and forwarded for AD processing. Therefore, the instant t_n when SQF forwards the n-th packet to the Server is initiated as $t_0 = a_0$. For a constant parameter D that needs to be selected, it is given by the following recursive expression:

$$t_{n+1} = \max(t_n + D, a_{n+1}),\ n \geq 0, \tag{2}$$

$$hence:\ t_{n+1} - t_n \geq D\ . \tag{3}$$

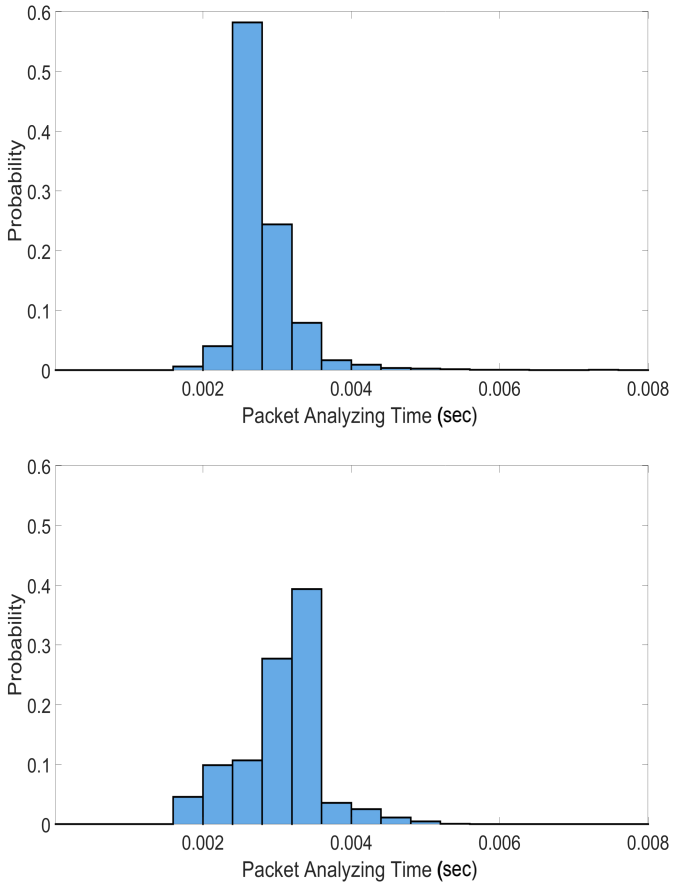

Fig. 9. The AD Processing Time per packet is shown at the Server when the SQF is used, and $D = 2.7\ ms$. The average AD packet processing time is $2.97\ ms$ and its variance is $0.0041\ sec^2$ when the attack occurs (above). The consequence of the Flood Attack (below) is to increase the AD average processing time of the AD per packet by just 10% to $3.28\ ms$, and a variance of $0.0023\ sec^2$. This again demonstrates the effectiveness of the SQF.

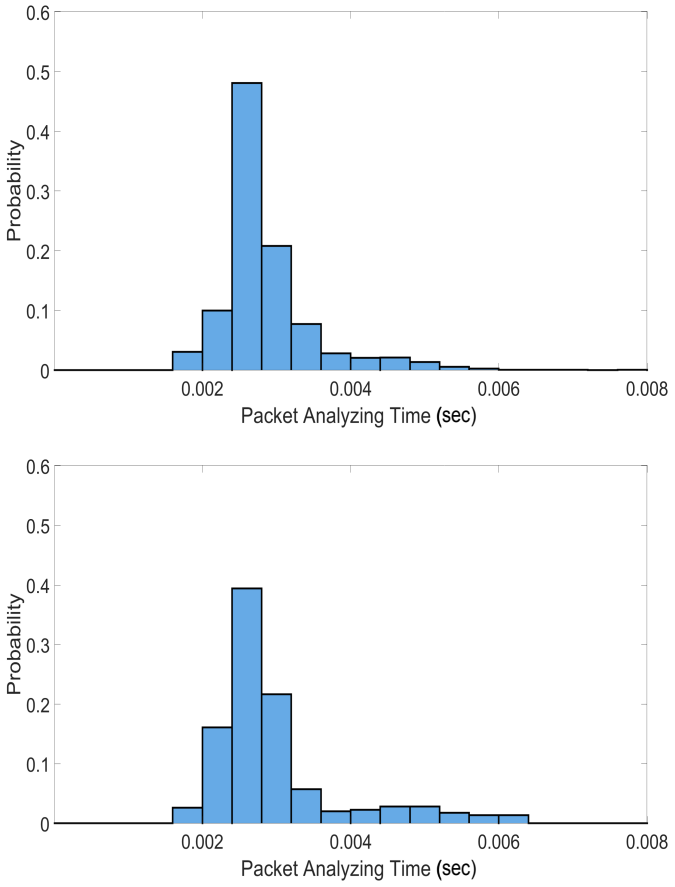

Fig. 10. AD Processing Time at the Server with the SQF and $D = 3.20\ ms$, resulting in an average AD processing time T_n of $3.00\ ms$ with a variance of $0.0036\ sec^2$ when there is no attack (above). Under a Flood Attack (below), the average AD processing time remains nearly the same at $2.99\ ms$ with a variance of $0.0067\ sec^2$. This again shows that SQF is very effective in avoiding Server slowdown during an attack.

The total delay Q_n experienced by the n-th packet due to the SQF, elapsing from the arrival of the n-th packet to the SQF at a_n, until its arrival to the AD at the Server at t_n, is then:

$$\begin{aligned} Q_0 &= t_0 - a_0 = 0, \qquad (4)\\ Q_{n+1} &= t_{n+1} - a_{n+1},\\ &= \max(t_n + D, a_{n+1}) - a_{n+1},\\ &= 0,\ if\ t_n + D \le a_{n+1},\ and\\ &= t_n + D - a_{n+1},\ otherwise. \qquad (5) \end{aligned}$$

Since $t_n = Q_n + a_n$, we obtain the recursive expression:

$$\begin{aligned} Q_{n+1} &= \max(0, t_n + D - a_{n+1}),\\ &= \max(0, Q_n + D - A_{n+1}),\ n \ge 0, \qquad (6) \end{aligned}$$

which is also an instance of Lindley's equation (1).

On the other hand, the Server's AD module also acts as an FCFS queue and we can exploit Lindley's equation again to compute $W_n,\ n \ge 0$ the waiting time of the n-th packet that arrives at the Server to be processed for attack detection,

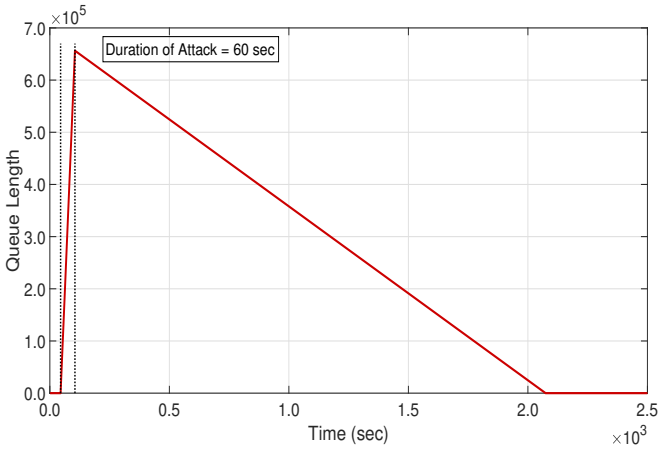

Fig. 11. SQF queue length (y-axis in the number of packets) against time (x-axis in seconds) when a UDP Flood Attack lasts 60 seconds, with $D = 3\ ms$, and without any mitigation action.

which is:

$$\begin{aligned} W_{n+1} &= \max(0, W_n + T_n - (t_{n+1} - t_n)),\ W_0 = 0,\\ &\le W_n + T_n - (t_n - t_{n+1}), \qquad (7) \end{aligned}$$

since the n-th packet's AD service time is T_n and the $n+1$-th interarrival interval to the Server's AD queue is $t_{n+1} - t_n$. Therefore, from (3) and (7) we have:

$$W_{n+1} \leq W_n + T_n - D, \tag{8}$$

and we obtain the following key result which tells us how to choose D:

Result 1. If D in the SQF is chosen to satisfy the inequality $D > T_n$ for any $n \geq 0$, then W_n, the packet waiting time at the Server, will be $W_n = 0, \ \forall \ n \geq 0$.

A. Experiments that Illustrate the Value of Result 1

Since the data in Figure 4 (above) shows the histogram of the AD processing time per packet T_n with average value 2.98 ms when there is no attack, we set $D = 3$ ms, just above that average as indicated by Result 1.

The experimental results in Figure 7 illustrate the case **without SQF** (above) and **with SQF** (below) during a 60 sec UDP Flood Attack. Note that the figure above represents the Server queue length varying over time, without the SQF. The figure below shows the Server queue length with a logarithmic scale, and compares the cases without SQF (in red) and with SQF (in blue) for the Server queue length that varies over time. Since $D = 3$ ms is very close to the average of T_n, the fluctuations in the values of T_n cause the buildup of a short queue of a few packets, as seen in the blue plot shown below.

Figure 8 shows the results of four experiments where we measure the queue length at the Server when a UDP Flood Attack lasts 30 (above) and 10 (below) seconds, without (red) and with (blue) the SQF. Without the SQF, the Server's AD processing time increases significantly. In the 30 sec attack, approximately $470,000$ packets are received at the Server and without the SQF it takes 44.45 minutes for the Server to return to normal process them, while in the 10 sec attack $153,667$ packets are received, and it takes the Server roughly 15 minutes to process them. Note that in these curves, it takes some 99 seconds for the compromised RPi to launch the attack.

Figure 9 shows that when we use the SQF based system with $D = 2.7$ ms, which is smaller than the value recommended by Result 1, when there is no attack, this choice of D has very little effect. However, when a UDP Flood Attack occurs, the Server's AD processing is somewhat slowed down, and the average value of T_n increases by roughly 10%.

On the other hand, Figure 10 confirms **Result 1** since it shows that if we take $D = 3.2$ ms, which guarantees that $D > T_n$ most of the time, then the measured average value of T_n remains at around 3 ms showing that it has not been slowed down by the attack's overload effect. Of course, the same is seen when no attack occurs.

1) SQF Queue Buildup and Attack Mitigation: During a Flood Attack, packets accumulate in the SQF input queue. The QDTP algorithm forwards them to the Server with $D = 3$ ms, and we observe that the Server does not experience any significant slowdown with regard to AD, as shown in Figure 11. We note the sudden queue length increase to over $600,000$ packets, followed by a very slow queue length decrease during more than $2,000$ secs, for a Flood Attack that only lasts 60 secs.

This motivates us to develop the novel, and hitherto unpublished, Adaptive Attack Mitigation (AAM) method that is discussed in Section IV.

IV. Adaptive Attack Mitigation (AAM)

In the previous Section, we observed that the SQF allows the AD to operate effectively during an attack by limiting the input packet rate. However, the SQF does not stop the huge build-up of attack packets at the input of the SQF, where the queue length increases to a very high value, as seen in Figure 11. Without an effective mitigation scheme, these packets will have to be processed long after the attack itself may have stopped, even though they are largely attack packets that are of no use to the system. Thus, in this Section, we propose a novel Adaptive Attack Mitigation (AAM) scheme, which will:

- Reduce the amount of AD testing that is carried out during an attack, and hence reduce the computational overhead at the Server,
- Drop attack packets and reduce the effect of traffic congestion and overload both during and after an attack,
- Stop the DROP process in a timely fashion when the attack ends, to avoid the excessive loss of benign packets.

A. Attack Detection AD

Recall (again) that the AAM uses an AD based on the Auto-Associative Deep Learning Random Neural Network described in [14]. It was evaluated with the well-known Kitsune attack dataset [70], [71], yielding the highly accurate results shown in Figure 12.

The AD reaches a decision based on a Window of $W > 0$ successive packets that are tested sequentially. If the AD system detects that a majority of the W packets are of "ATTACK" type, then it concludes that an attack is occurring. Otherwise, it will return a NO-ATTACK decision. If a NO-ATTACK is detected, then the AD system will proceed to test the subsequent W packets in the same way. Note that W is chosen empirically to be the smallest value of the window that provides high accuracy, and is typically set in the range of 8 to 10 packets.

When the AD detects an ATTACK, since the AD is substantially slowed down by the large packet backlog that accumulates during the attack, the novel AAM technique that we propose, drops the preceding $m + W$ packets, and then skips ahead $m > 0$ packets in the incoming packet stream, reaching a subsequent Testing Window of W packets. Thus, the packet backlog is reduced, and the processing slowdown for AD is also reduced, since each individual packet is only tested by the AD when the system is **not** under attack.

Thus, AAM has the advantage of early attack detection, rapid dropping of blocks of attacking packets, and reducing the AD processing time during an attack by carrying out AD in successive m-packet intervals using a W-packet window. The AAM algorithm is detailed below:

1) Initialize $i \leftarrow 1$ and $j \leftarrow 1$.
2) Test the W packets $i, i+1, \ \ldots \ , i+W-1$ for AD.

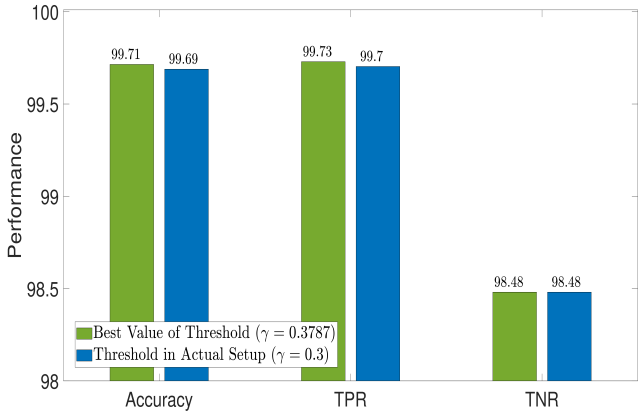

Fig. 12. Accuracy of the AADRNN attack detector, that was evaluated on the test-bed of Figure 1.

3) If the output of the AD is **ATTACK then**:
   - **DROP the PACKETS** $j, j+1, \ \ldots\ , i+W-1$,
   - **Update** $j \leftarrow i+W$ *and* $i \leftarrow i+W-1+m$
4) If the output of the AD is **NO-ATTACK then**:
   - **FORWARD Packets** $j, j+1, \ \ldots\ , i+W-1$ **to the SERVER**,
   - **Update** $j \leftarrow i+W$ *and* $i \leftarrow i+W$,
5) **Go To 2)**

The summarized pseudocode and the detailed pseudocode for the AAM are given below as **Algorithm 1** and **Algorithm 2**.

Algorithm 1 Summarized Pseudocode for the AAM

Initialize $i \leftarrow 1$, $j \leftarrow 1$
Let X be the total number of packets received during the attack
Let f be the fraction of attack packets
Let α and β be weighting factors for the cost function
Compute the initial optimal value of m:

$$m^* \leftarrow \sqrt{2\frac{\beta}{\alpha} \cdot W(E[X]-W)} - W$$

while there are incoming packets to process **do**
    Test the W (window) packets: $i, i+1, \ldots, i+W-1$ for Attack Detection
    if Attack Detection output is **ATTACK then**
        Drop the packets $j, j+1, \ldots, i+W-1$
        Update $j \leftarrow i+W$ and $i \leftarrow i+W-1+m^*$
        Update the total number of dropped packets.
        Calculate the expected value of the dropped benign packets cost.
        Update the total average cost of AAM.
    else ▷ If Attack Detection output is **NO-ATTACK**
        Forward the packets $j, j+1, \ldots, i+W-1$ to the server
        Update $j \leftarrow i+W$ and $i \leftarrow i+W$
        Update the total average cost of AAM.
    end if
    Recalculate m^* periodically as needed.
end while

Algorithm 2 Detailed Pseudocode for the AAM

Initialize $i \leftarrow 1$, $j \leftarrow 1$
Let X be the total number of packets received during the attack
Let f be the fraction of attack packets
Let α and β be weighting factors for the cost function
Compute the initial optimal value of m:

$$m^* \leftarrow \sqrt{2 \cdot \frac{\beta}{\alpha} W(E[X]-W)} - W$$

while there are incoming packets to process **do**
    Test the W (window) packets: $i, i+1, \ldots, i+W-1$ for Attack Detection
    if Attack Detection output is **ATTACK then**
        Drop the packets $j, j+1, \ldots, i+W-1$
        Update $j \leftarrow i+W$ and $i \leftarrow i+W-1+m^*$
        Update the total number of dropped packets:

$$\delta \leftarrow N(m+W)$$

        Calculate the expected value of the reprocessing cost dropped benign packets:

$$E[K] \approx W \cdot \tau \cdot \left[\frac{(1-f)E[X]}{W} - \frac{1}{2} + \frac{m}{2W}\right]$$

        Update the total average cost of AAM:

$$C(AAM) = \alpha \cdot E[K] + \beta \cdot E[\Omega]$$

        where

$$E[\Omega] \approx \tau \cdot W \cdot \left[\frac{E[X]-W}{m^*+W} + \frac{1}{2}\right]$$

    else ▷ If Attack Detection output is **NO-ATTACK**
        Forward the packets $j, j+1, \ldots, i+W-1$ to the server
        Update $j \leftarrow i+W$ and $i \leftarrow i+W$
        Update the total average cost of AAM.
    end if
    Recalculate m^* periodically as needed.
end while

B. Analysis and Optimization of AAM

During a Flood Attack, a large fraction, say $0 < f \leq 1$, of the incoming traffic will be part of the attack, and the complementary fraction $(1-f)$ will be benign traffic coming from various sources that also send traffic to the IoV Gateway. Thus, the AAM may drop useful benign packets, as well as attack packets.

In addition, even though the AAM reduces the number of packets that are actually tested for AD during an attack, the proposed AAM still creates computational overhead during an attack, because it tests W packets after each m packet interval. Interestingly, the proportion of benign packets which are dropped will increase as m increases, but at the same time the overhead also decreases as m increases. Thus, we will now compute the optimum value of m. In the next subsection, we will validate these analytical results through some experiments.

Let X denote the total number of packets received at the Gateway during an attack, including a fraction f of attack packets. Since X cannot be known in advance, we treat it here as a random variable. Note that we will denote the expected value of the random variable Y by $E[Y]$.

During the attack, there will be a first inevitable W-packet AD window when the attack is first detected. This first testing window where the AD says "ATTACK" will contain a majority of attack packets among the W packets. Since W is very small compared to X, we can assume that the attack begins at the beginning of the first testing window where the AD reports an attack.

The end of the attack is signaled to the AAM by the AD during the first detection window in which a majority of the W packets are not attack packets, which indicates that the current attack has ended. Thus, in addition to the first attack detection window, the total number of additional attack detection windows that are used during an attack is given by:

$$\begin{aligned} N &= \lceil \frac{X-W}{m+W} \rceil, \ and \\ E[N] &\approx \frac{E[X]-W}{m+W} + \frac{1}{2}, \end{aligned} \tag{9}$$

where the expression (9) is actually a mathematically proven [72] first-order approximation. In particular, it applies to all probability density functions for the random variable X which are a convex sum of Erlang densities, commonly used to approximate discrete histograms.

If the AAM is used in conjunction with the SQF that includes the QDTP traffic shaping policy, then Figure 10 shows that the AD average processing time per packet remains constant at roughly 3 $msec$ which we will call τ, which varies with the speed of the processor at the Server. For each detection window of W packets, the Server overhead is then $N\tau W$, so that the resulting overhead for attack detection when an attack occurs is:

$$\Omega = N\tau W, \ with \ E[\Omega] \approx \tau W[\frac{E[X]-W}{m+W} + \frac{1}{2}]. \tag{10}$$

On the other hand, the total number of dropped packets due to the attack is:

$$\begin{aligned} \delta &= W + N \times m + (N-1) \times W = N(m+W), \\ E[\delta] &\approx E[X] + \frac{1}{2}(m-W), \end{aligned} \tag{11}$$

since the packets in the first window that says "ATTACK", as well as the following m packets are dropped. This is repeated a total of N times, while those in the packets in the last window that say "NO-ATTACK" are not dropped.

However, the part X of δ includes attack and benign packets. Let the fraction of benign packets in X be $(1-f)$. Since we know that the last part $\delta - X$ is only composed of benign packets, the reprocessing time K of **the benign packets which the AAM drops** may create additional overhead, since it is likely that they will be sent again to the Server after some length of time, and the Server will have to process them again using the AD. Hence:

$$\begin{aligned} K &= \tau W \lceil \frac{(1-f)X + \delta - X}{W} \rceil, \ and \\ E[K] &\approx \tau W [\frac{(1-f)E[X]}{W} - 1 + \frac{m+W}{2W}] \ , \\ &\approx \tau W [\frac{(1-f)E[X]}{W} - \frac{1}{2} + \frac{m}{2W}] \ , \end{aligned} \tag{12}$$

and the total average cost of AAM written as $C(AAM)$, assuming a weighting of $\alpha, \ \beta > 0$ for the two average cost terms $E[K]$ and $E[\Omega]$, respectively, becomes:

$$\begin{aligned} &C(AAM) = \alpha E[K] + \beta E[\Omega] \\ &\approx W\tau \big[\ \alpha [\frac{(1-f)E[X]}{W} - \frac{1}{2} + \frac{1}{2}\frac{m}{W}] \\ &+\beta [\frac{E[X]-W}{m+W} + \frac{1}{2}] \big]. \end{aligned} \tag{13}$$

Thus, taking the derivative of the right-hand-side of (13) with respect to m, and setting it to zero, we can see that the **total average cost** $C(AAM)$ is **approximately minimized by setting** m to the value m^*:

$$m^* \approx \sqrt{2\frac{\beta}{\alpha} W [\ E[X] - W \ ]} \ - W, \tag{14}$$

where $\frac{\beta}{\alpha}$ is the relative importance of the two terms in the cost function; $\beta << \alpha$ when the benign packets that were dropped during an attack arrive (for the second time) at the Server while it is busy processing other incoming benign packets.

Interestingly, we see that m^* does **not depend on** f and τ. Furthermore, as the average number of packets $E[X]$ received by the Server during the attack increases, the optimum value m^* also increases in proportion to the square root of $E[X]$.

C. Scalability

We now discuss the scalability of the Optimum AAM Algorithm. The scalability will be discussed in the context of a Gateway Server with multiple network ports, since the AD and the Optimum AAM are designed to protect a **single** Gateway Server. Let us first recall that the choice of the parameter W is discussed in Section IV-A, and that it is chosen empirically as the smallest value that provides sufficiently high accuracy for the AD algorithm, typically around $W \approx 8 \ to \ 10$ consecutive packets.

For a single Gateway Server with P network ports, we can use the Optimum AAM in several ways: (a) we may have one AD and one AAM for each port, or (b) one common AD and then one AAM per port, or (c) one AD for each port and a single common AAM, or (d) one AD and one single AAM for all the ports. In all these cases the value of W will be the same.

Notice that the total processing time of all the ADs is proportional to the total number of packets that they process; thus the AD processing time is not affected by the configurations (a)-(d), but simply by the total traffic entering the Gateway from all ports; if all ports carry an equal amount of traffic, then the total AD overhead processing time will increase linearly with P. However, in the cases (b) and (d)

where there is a single AD for all ports,in order to determine which port is being attacked, the AD will have to group incoming packets into W-windows of packets with the same incoming port number, which requires additional computation. Therefore, with regard to the AD processing time, the solutions (a) and (c) are more efficient than (b) and (d).

Similarly, the AAM is only activated when an attack is detected, so that the total number of AAM activations which cause processing overhead is the same in all cases and depends on the total number of attacks that aim the Gateway Server, rather than on the number of ports. However, one may argue that more ports will attract more attacks. On the other hand, having a single Optimum AAM unit, as in cases (c) and (d), has two shortcomings: (i) when multiple attacks occur within short intervals, a backlog of processing requests may accumulate in front of the single AAM software unit, leading to delays in the mitigation process, and (ii) the single AAM will have to deal with multiple distinct values of m^*, leading to sub-optimal decisions. Thus we recommend the use of (a) for the best performance and responsiveness to attacks.

D. Optimization Measurements of the AAM Scheme

In this subsection, we present the measurements of the proposed AAM scheme. Figure 13 illustrates the theoretically computed average total cost $C(AAM)$, across various parameter values when $W = 20$, $\alpha = 20\beta$, and we vary the value of $E[X]$. We have run experiments where we first generate the value of X, i.e. the number of packets in an attack against the Gateway Server, at random. These experiments are run thirty times for each fixed value of the average number of packets $E[X]$ received during a Flood Attack. The experiments are also run for several different values of $E[X]$. In each experiment, we compute the corresponding value of the cost function $C(AAM)$ that results from the attack and compute its average value by taking the average of the cost values obtained from the distinct measurements for a given $E[X]$.

These experimental results are summarized in Figure 14. They provide an empirical demonstration of the accuracy and validity of the theoretical formula (14) which predicts the value of m^*.

E. The Effect of the AAM on the Gateway's Time-Line Behaviour during Attacks

We now measure the time dependent queue length of packets at the entrance of the SQF during two successive Flood Attacks. The first attack involves over $10,000$ packets, and the second one is comprised of some $40,000$ packets. The AD system processes the incoming packet stream and provides an ATTACK alert, which is sent to the AAM. The AAM then measures the SQF input queue length, and uses the formula (14), to choose m^*, which is 127 for the first attack, and 248 for the second one.

The effectiveness of the AAM is demonstrated in Figure 15, by the speed with which the AAM drops packets after each of the two successive attacks, and reacts to the second attack in a timely fashion despite the first attack.

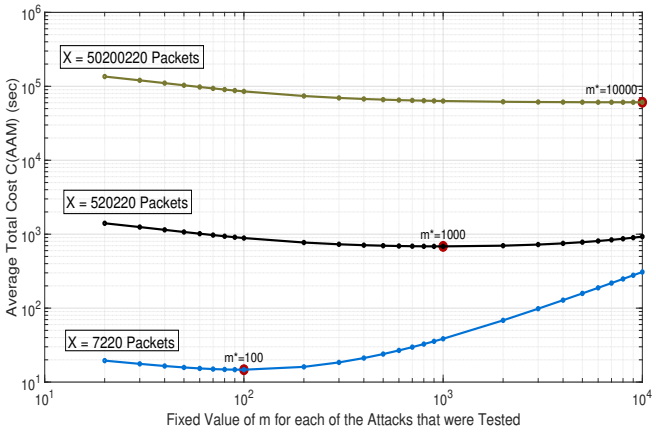

Fig. 13. Theoretical graph of the Average Total Cost plotted against the parameter m when both the x and y axes are logarithmic for $\frac{\beta}{\alpha} = 0.05$. We show how the Average Cost $C(AAM)$ depends on m and exhibit the theoretical minimum Average Total Cost at the value m^*, which is obtained analytically.

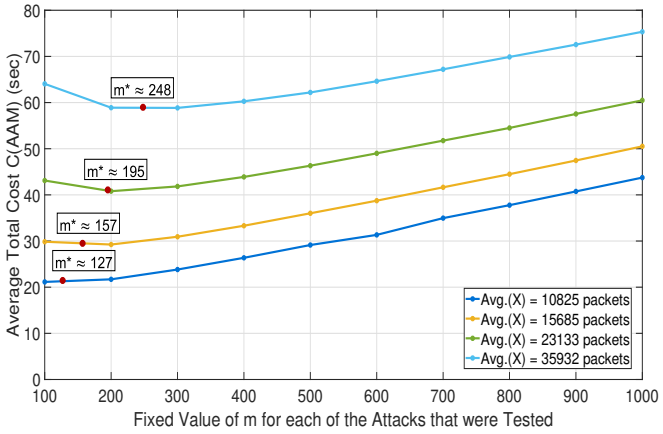

Fig. 14. Experimental graph (when both the x and y coordinates are linear, i.e. not logarithmic) of the Average Total Cost $C(AAM)$ over thirty independent experiments for $\frac{\beta}{\alpha} = 0.05$, plotted against the parameter m. We see that the average cost depends on m, and confirms that the theoretically predicted m^* provides a useful value of cost minimization.

For the two attacks that were shown in Figure 15, the time dependent packet queue length at the entrance of the AD system is shown in Figure 16. This latter figure also demonstrates that the joint use of the SQF and the AMM, is able to effectively limit the AD input queue to a very small number of 20, very rapidly after the attack begins; thus, the AD system is able to operate continuously and effectively, contributing to the correct decisions that are being made by the AAM.

V. Conclusions

In this research, we have first studied the impact of UDP Flood Attacks on an IoV Gateway Server that supports an AD system. We demonstrate that even short-duration attacks can cause significant overload for the Sever. the Gateway Server. As a result, The Server's normal operations, including AD, are substantially slowed down. Indeed, we observe that an attack

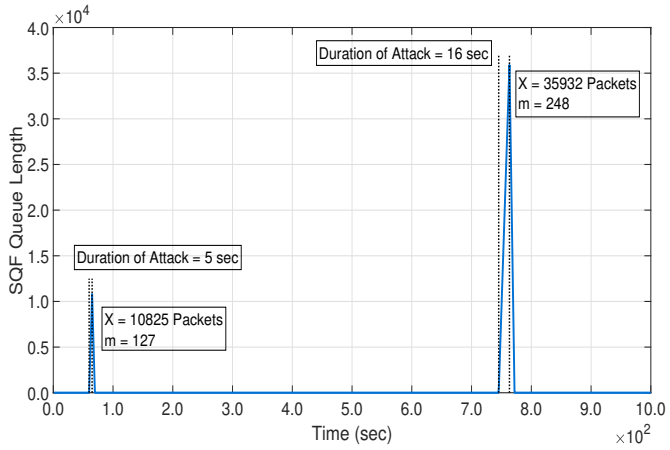

Fig. 15. Time-line of queue length measurements at the entrance of the SQF for two successive attacks. The first attack involves around $10,000$ packets, while the second one is more severe and involves close to $40,000$ packets. The AD processes the incoming traffic from the SNMP in batches of W packets, and provides an attack alert when an attack is detected. AAM then goes into action, observes the packet lengths, and uses the formula (14) to select the optimum value m^*, which is 127 in the first case, and 248 in the second case. We observe the effectiveness of the AAM in rapidly dropping attack packets, and also reacting to the second attack in a timely fashion after the first attack is cleared.

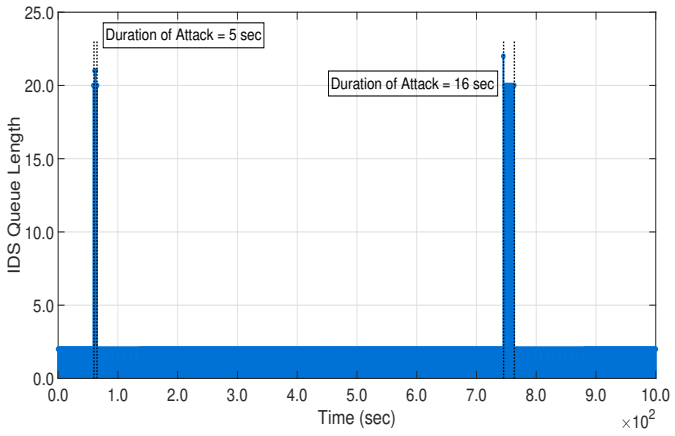

Fig. 16. Time-line of queue length measurements at the entrance of the AD system, when the two successive attacks shown in Figure 15 occur. Despite the fact that the two successive attacks involve $10,000$ and $40,000$ packets, the conjunction of the use of the SQF and the AAM, limits the AD input queue to around 20 packets for a very short period, without overwhelming the AD and allowing it to operate smoothly and continuously.

that lasts 60 seconds may create a backlog of packets at the Server that requires several hours to clear out.

Thus, we propose that the Gateway Server's input should be "protected" by a special SQF front end that operates the QDTP policy, to allow the timely operation of the Server even when an attack occurs. This approach requires installing an inexpensive lightweight hardware addition, such as an RPi, between the local area network that supports the Sensors, and the IoV Gateway Server.

Several experiments are then used to illustrate the effectiveness of the proposed approach, and we note that the SQF front end requires that a key timing parameter D be chosen. We provide a theoretical analysis of how D should be selected, and show that it must be comparable but larger than the AD processing time per packet under non-attack conditions. We validate this theoretical observation with several experiments and show that the SQF effectively preserves the Gateway Server from congestion and overload, allowing it to operate normally, even in the presence of severe Flood Attacks.

However, we note that the congestion that is eliminated at the Server may now accumulate at the SQF input, creating excessive packet processing work for the Gateway. Therefore, we propose the novel mitigation action AAM which is activated when the AD detects an attack. The AAM is able to drop incoming attack packets in a relatively short time, and is designed to minimize a cost function that combines the number of dropped benign packets and the overhead of testing the incoming packet stream for AD to identify the end of the attack. The AAM is tested experimentally and shown to be very effective during significant Flood Attacks.

While this paper has focused on an architecture with multiple sources of IoV traffic and a single Gateway, future work will consider Edge Systems with multiple devices and Gateways. We plan to study dynamic policies for AD for complex IoV networks having both static and mobile nodes, with mitigation techniques that include traffic routing and packet drops at the Edge.

The energy consumption of such systems is another important issue that we plan to address, so that dynamic management policies may minimize energy consumption, optimize system Quality of Service, and offer Cybersecurity.

REFERENCES

[1] Cisco, *Cisco Annual Internet Report (2018–2023)*, Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] J. Liu, L. Song *et al.*, "A novel congestion reduction scheme for massive machine-to-machine communication," *IEEE Access*, vol. 5, pp. 18 765–18 777, 2017.

[3] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, and L. Guijarro, "Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3505–3520, 2018.

[4] E. Johns and M. Ell, "Cyber security breaches survey 2023," April 2023. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023

[5] Cloudflare, 'Famous DDoS attacks: The largest DDoS attacks of all time," 3 Jan. 2025 [Online]. Available: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/

[6] E. Gelenbe and Y. M. Kadioglu, "Performance of an autonomous energy harvesting wireless sensor," in *Information Sciences and Systems 2015*, London, September 22-24, 2015, O. H. Abdelrahman, E. Gelenbe, G. Gorbil, and R. Lent, Eds. Cham: Springer International Publishing, 2016, pp. 35–43.

[7] E. Gelenbe, "A diffusion model for packet travel time in a random multihop medium," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 2, pp. 1-10, 2007.

[8] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *EEE Transactions on Communications*, vol. 68, no. 8, p. 4734–4746, 2020.

[9] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, no. 5, p. 4641–4654, 2020.

[10] J. Voelcker, "1.2 billion vehicles on world's roads now, 2 billion by 2035," *Green Car News*, July 2014. [Online]. Available: https://www.greencarreports.com/news/1093560_1-2-billion-vehicles-on-worlds-roads-now-2-billion-by-2035-report

[11] N. M. Gökhan, "Development of a simultaneous design for supply chain process for the optimization of the product design and supply chain configuration problem," *Engineering Management Journal*, vol. 2, no. 4, p. 20–30, 2010.

[12] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, p. 997–1002.

[13] A. Frötscher, et al., "Improve cybersecurity of C-ITS Road Side Infrastructure Installations: the SerIoT - secure and safe IoT approach," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, 2019, pp. 1–5.

[14] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against iot-connected home environments," *Procedia Computer Science*, vol. 134, pp. 458–463, 2018.

[15] E. Gelenbe and M. Nakıp, "Traffic based sequential learning during botnet attacks to identify compromised IoT devices," *IEEE Access*, vol. 10, pp. 126 536–126 549, 2022.

[16] M. Nakip and E. Gelenbe, "Online self-supervised deep learning for intrusion detection systems," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 5668–5683, 2024.

[17] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Computers & Security*, vol. 103, p. 102150, 2021.

[18] S. T. Banafshehvaragh and A. M. Rahmani, "Intrusion, anomaly, and attack detection in smart vehicles," *Microprocessors and Microsystems*, vol. 96, p. 104726, 2023.

[19] M. Girdhar, J. Hong, and J. Moore, "Cybersecurity of autonomous vehicles: A systematic literature review of adversarial attacks and defense models," *IEEE Open Journal of Vehicular Technology*, 2023.

[20] D. Man, F. Zeng, J. Lv, S. Xuan, W. Yang, and M. Guizani, "Ai-based intrusion detection for intelligence internet of vehicles," *IEEE Consumer Electronics Magazine*, vol. 12, no. 1, pp. 109–116, 2021.

[21] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer networks*, vol. 44, no. 5, pp. 643–666, 2004.

[22] G. Oke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with bayesian classifiers and the random neural network," in *2007 IEEE International Fuzzy Systems Conference*, 2007, pp. 1–6.

[23] H. Ahmetoglu and R. Das, "A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions," *Internet of Things*, vol. 20, p. 100615, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S254266052200097X

[24] M. Banerjee and S. Samantaray, "Network traffic analysis based iot botnet detection using honeynet data applying classification techniques," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 8, 2019.

[25] T. A. Tuan, H. V. Long, R. Kumar, I. Priyadarshini, N. T. K. Son *et al.*, "Performance evaluation of Botnet DDoS attack detection using machine learning," *Evolutionary Intelligence*, pp. 1–12, 2019.

[26] A. I. Al-issa, M. Al-Akhras, M. S. ALsahli, and M. Alawairdhi, "Using machine learning to detect DoS attacks in wireless sensor networks," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 107–112.

[27] E. Y. Güven and Z. Gürkaş-Aydın, "Mirai botnet attack detection in low-scale network traffic," *Intelligent Automation & Soft Computing*, vol. 37, no. 1, pp. 419–437, 2023.

[28] H. Sedjelmaci, N. Kaaniche, A. Boudguiga, and N. Ansari, "Secure attack detection framework for hierarchical 6g-enabled internet of vehicles," *IEEE Transactions on Vehicular Technology*, 2023.

[29] L. Yang and A. Shami, "A transfer learning and optimized cnn based intrusion detection system for internet of vehicles," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 2774–2779.

[30] M. Aloqaily, S. Otoum, I. Al Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Networks*, vol. 90, p. 101842, 2019.

[31] S. Aurangzeb, M. Aleem, M. T. Khan, H. Anwar, and M. S. Siddique, "Cybersecurity for autonomous vehicles against malware attacks in smart-cities," *Cluster Computing*, pp. 1–16, 2023.

[32] F. Pascale, E. A. Adinolfi, S. Coppola, and E. Santonicola, "Cybersecurity in automotive: An intrusion detection system in connected vehicles," *Electronics*, vol. 10, no. 15, p. 1765, 2021.

[33] L. Yang, A. Moubayed, and A. Shami, "Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2021.

[34] E. Eziama, F. Awin, S. Ahmed, L. Marina Santos-Jaimes, A. Pelumi, and D. Corral-De-Witt, "Detection and identification of malicious cyber-attacks in connected and automated vehicles' real-time sensors," *Applied Sciences*, vol. 10, no. 21, p. 7833, 2020.

[35] J. Mirkovic, S. Fahmy, P. Reiher, and R. K. Thomas, "How to test dos defenses," in *2009 cybersecurity applications & technology conference for homeland security*. IEEE, 2009, pp. 103–117.

[36] M. Kaouk, F.-X. Morgand, and J.-M. Flaus, "A testbed for cybersecurity assessment of industrial and IoT-based control systems," in *Lambda Mu 2018 - 21è Congrè de Maîtrise des Risques et Sûreté de Fonctionnement, Oct 2018, Reims, France*. [Online]. Available: https://hal.science/hal-02074654v1/document

[37] M. Annor-Asante and B. Pranggono, "Development of smart grid testbed with low-cost hardware and software for cybersecurity research and education," *Wireless Personal Communications*, vol. 101, pp. 1357–1377, 2018.

[38] O. A. Waraga, M. Bettayeb, Q. Nasir, and M. A. Talib, "Design and implementation of automated iot security testbed," *Computers & security*, vol. 88, p. 101648, 2020.

[39] V. K. Singh, R. Sharma, and M. Govindarasu, "Testbed-based performance evaluation of attack resilient control for wind farm scada system," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2020, pp. 1–5.

[40] A. Ghaleb, S. Zhioua, and A. Almulhem, "Scada-sst: a scada security testbed," in *2016 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2016, pp. 1–6.

[41] A. Tesfahun and D. L. Bhaskari, "A scada testbed for investigating cyber security vulnerabilities in critical infrastructures," *Automatic Control and Computer Sciences*, vol. 50, pp. 54–62, 2016.

[42] S.-U. Park and S.-M. Hwang, "Test bed construction for apt attack detection," *International Journal of Control and Automation*, vol. 11, no. 4, pp. 175–186, 2018.

[43] R. Arthi and S. Krishnaveni, "Design and development of iot testbed with ddos attack for cyber security research," in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. IEEE, 2021, pp. 586–590.

[44] A. P. Wright and N. Ghani, "A testbed for the evaluation of denial of service attacks in software-defined networks," in *2019 SoutheastCon*. IEEE, 2019, pp. 1–6.

[45] P. V. Sontakke and N. B. Chopade, "Impact and analysis of denial-of-service attack on an autonomous vehicle test bed setup," in *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2021*. Springer, 2022, pp. 221–236.

[46] Q. He, X. Meng, R. Qu, and R. Xi, "Machine learning-based detection for cyber security attacks on connected and autonomous vehicles," *Mathematics*, vol. 8, no. 8, p. 1311, 2020.

[47] T. H. Aldhyani and H. Alkahtani, "Attacks to automatous vehicles: A deep learning algorithm for cybersecurity," *Sensors*, vol. 22, no. 1, p. 360, 2022.

[48] A. Kavousi-Fard, M. Dabbaghjamanesh, T. Jin, W. Su, and M. Roustaei, "An evolutionary deep learning-based anomaly detection model for securing vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4478–4486, 2020.

[49] I. Ahmed, G. Jeon, and A. Ahmad, "Deep learning-based intrusion detection system for internet of vehicles," *IEEE Consumer Electronics Magazine*, vol. 12, no. 1, pp. 117–123, 2021.

[50] P. Sharma and H. Liu, "A machine-learning-based data-centric misbehavior detection model for internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4991–4999, 2020.

[51] T. Alladi, V. Kohli, V. Chamola, and F. R. Yu, "Securing the internet of vehicles: A deep learning-based classification framework," *IEEE networking letters*, vol. 3, no. 2, pp. 94–97, 2021.

[52] E. Gelenbe, B. C. Gul, and M. Nakip, "Disfida: Distributed self-supervised federated intrusion detection algorithm with online learning for health internet of things and internet of vehicles," *Internet of Things*, vol. 28, no. https://doi.org/10.1016/j.iot.2024.10134, p. 101340, 2024.

[53] T. Alladi, V. Kohli, V. Chamola, F. R. Yu, and M. Guizani, "Artificial intelligence (ai)-empowered intrusion detection architecture for the internet of vehicles," *IEEE Wireless Communications*, vol. 28, no. 3, pp. 144–149, 2021.

[54] Y. Sun, L. Wu, S. Wu, S. Li, T. Zhang, L. Zhang, J. Xu, Y. Xiong, and X. Cui, "Attacks and countermeasures in the internet of vehicles," *Annals of Telecommunications*, vol. 72, pp. 283–295, 2017.

[55] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[56] X. Li, Z. Hu, M. Xu, Y. Wang, and J. Ma, "Transfer learning based intrusion detection scheme for internet of vehicles," *Information Sciences*, vol. 547, pp. 119–135, 2021.

[57] B. T. Alemu, A. J. Muhammed, H. M. Belachew, and M. Y. Beyene, "A comprehensive detection and mitigation mechanism to protect sd-iov systems against controller-targeted ddos attacks," *Cluster Computing*, vol. 27, no. 10, pp. 14 295–14 313, 2024.

[58] A. A. Almazroi, M. H. Alkinani, M. A. Al-Shareeda, and S. Manickam, "A novel ddos mitigation strategy in 5g-based vehicular networks using chebyshev polynomials," *Arabian Journal for Science and Engineering*, vol. 49, no. 9, pp. 11 991–12 004, 2024.

[59] U. Tariq, "Optimized feature selection for ddos attack recognition and mitigation in sd-vanets." *World Electric Vehicle Journal*, vol. 15, no. 9, 2024.

[60] Z. Li, Y. Kong, C. Wang, and C. Jiang, "Ddos mitigation based on space-time flow regularities in iov: A feature adaption reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2262–2278, 2021.

[61] A. Haydari and Y. Yilmaz, "Rsu-based online intrusion detection and mitigation for vanet," *Sensors*, vol. 22, no. 19, p. 7612, 2022.

[62] E. Gelenbe and M. Nasereddin, "Protecting IoT servers against flood attacks with the quasi deterministic transmission policy, (Best Paper Award, IEEE Trustcom 2023)," in *22nd IEEE International Conference on Trust, Security and Privacy Computing and Communications, November 2023, Exeter, UK*, vol. 2023, no. https://arxiv.org/pdf/2306.11007.pdf, 2024, pp. 1–8.

[63] J. Augusto-Gonzalez, et al., "From internet of threats to internet of things: A cyber security architecture for smart homes," in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2019, pp. 1–6.

[64] E. Gelenbe and K. Sigman, "Iot traffic shaping and the massive access problem," in *ICC 2022, IEEE International Conference on Communications, 16–20 May 2022, Seoul, South Korea*, no. https://zenodo.org/record/5918301. https://zenodo.org/record/5918301, 2022, pp. 1–6.

[65] M. Masirap, M. H. Amaran, Y. M. Yussoff, R. Ab Rahman, and H. Hashim, "Evaluation of reliable udp-based transport protocols for internet of things (iot)," in *2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. IEEE, 2016, pp. 200–205.

[66] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[67] ——, "G-networks with instantaneous customer movement," *Journal of Applied Probability*, vol. 30, no. 3, pp. 742–748, 1993.

[68] E. Gelenbe and Y. Yin, "Deep learning with dense random neural networks," in *International Conference on Man–Machine Interactions*, no. DOI:10.1007/978-3-319-67792-. Springer, Cham, 2017, pp. 3–18.

[69] "MHDDoS - DDoS Attack Script With 56 Methods," Online, May 2022, accessed: 2023-02-22. [Online]. Available: https://github.com/MatrixTM/MHDDoS

[70] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *The Network and Distributed System Security Symposium (NDSS) 2018*, 2018.

[71] "Kitsune Network Attack Dataset," August 2020. [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune

[72] E. Gelenbe, J. C. A. Boekhorst, and J. L. W. Kessels, "Minimizing wasted space in partitioned segmentation," *Commun. ACM*, vol. 16, no. 6, p. 343–349, jun 1973. [Online]. Available: https://doi.org/10.1145/362248.362253