Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

PETRAK: A solution against DDoS attacks in vehicular networks

Amandeep Verma [a,d], Rahul Saha [b,c,*], Gulshan Kumar [b,c], Mauro Conti [c]

[a] School of Computer Applications, Lovely Professional University, Punjab, India
[b] School of Computer Science and Engineering, Lovely Professional University, Punjab, India
[c] Department of Mathematics, University of Padua, Padua, Italy
[d] Centre of Research Impact and Outcome, Chitkara University, Rajpura 140417, Punjab, India

ARTICLE INFO

Keywords:
VANET
Machine learning
DDoS
Mitigation
Prevention
Detection

ABSTRACT

In recent years, the frequently reported incidents of Distributed Denial of Service assaults on vehicular networks in various countries have made researchers find new protective solutions. DDoS attacks can propagate through the charging points for electric vehicles in a charging station and affect the production of critical infrastructures such as electric grids. Existing solutions are efficient in attack detection; however, current systems do not offer multi-level protection, and zero-day vulnerabilities are prone to escape from the detection systems.

In this paper, we address the problems mentioned above and introduce the first Machine Learning (ML)–based DDoS protective solution to combine prevention and detection mechanisms in vehicular networks. To be more specific, our proposed model is the first to consider the adaptive traffic threshold to generate the alarm for a suspicious amount of traffic flow in an Intrusion Detection Prevention System (IDPS). We call our proposed approach *Protecting vEhicular neTworks against distRibuted deniAl of service attacKs (PETRAK)*. PETRAK uses four functions: prevention, alarm, training, and detection. The alarming system uses the flow parameters and activates the detection module to detect malicious packets. The prevention system works in two modes: immediate and future. PETRAK uses logistic regression to identify incoming packets and signatures of malicious packets to prevent future attacks. We also show that our proposed model is implacable towards advanced post-quantum cryptography-based traffic and also able to analyse side-channel attacks. We run a comprehensive set of experiments to test PETRAK on our synthesized dataset, KDDCUP’99 dataset, CIC-MalMem-2022, and the ToN-IoT dataset. We observe that PETRAK shows an accuracy of 99%. The results claim the efficiency of PETRAK in detecting and preventing DDoS attacks in vehicular networks.

1. Introduction

Cybersecurity breaches become an economic risk when we integrate information technology into businesses and industries. Firms and industries heavily depend on road, rail, and air transportation, and all these transportation mediums are always under cybersecurity threats. A report from CISCO finds 1272 critical intrusions until November 2019, with the exposure of around 163 million records. In 2019, the average number of documents exposed per data breach counted was 128,171 [1]. A set of malware-based assaults includes threats such as spam, viruses, botnets, phishing, hacktivist worms, spyware, spoofing, malware, and DDoS.

Attacks on rail and aviation networks’ security have become more frequent in recent years. The public transportation system in San Francisco faced data infiltration in November 2016 [2]. The passenger information system was shut down in 2017 due to a cyber attack on Germany’s train system [3]. The UK railway network was the victim of four cyber attacks between 2015 and 2016 [4]. A report of a recent attack on Italy’s train network in March 2022 impedes train movement and ticket sales at the station [5]. The Iranian train system has also faced a cybersecurity risk known as MeteorExpresss [6]. Belarusian hackers have chosen to target Russia’s train systems to obstruct the movement of Russian soldiers [6]. Malware is one of the weapons that is used by attackers to target vehicular networks. The increasing demand for vehicular applications increases the vulnerabilities in the vehicular networks. The existing attack detection models are unable to handle the new malware and adapt to the thresholds of vehicular traffic. Besides, it is also required to have a prevention system for vehicular networks, which is less researched in existing literature. Therefore, vehicular networks need an efficient security model to detect and prevent DDoS attacks for the sustainable growth of Intelligent Transport Systems (ITS).

* Corresponding author at: Department of Mathematics, University of Padua, Padua, Italy.
E-mail address: rsahaaot@gmail.com (R. Saha).

https://doi.org/10.1016/j.comcom.2024.04.025
Received 6 October 2023; Received in revised form 13 March 2024; Accepted 21 April 2024
Available online 25 April 2024
0140-3664/© 2024 Elsevier B.V. All rights reserved.

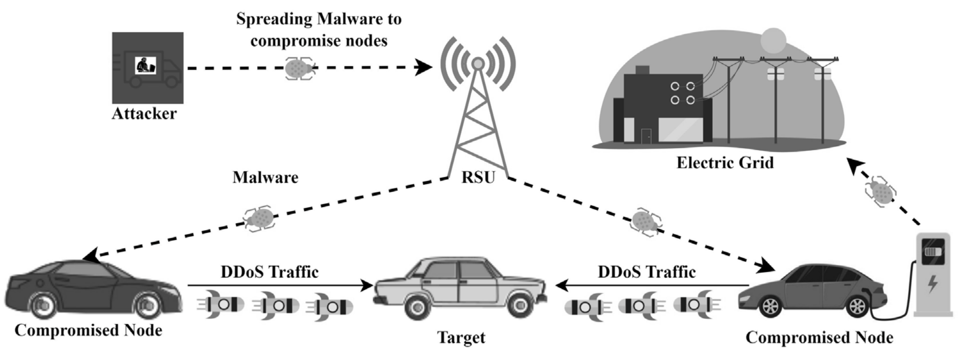

Fig. 1. Malware-based DDoS attack.

1.1. Malware-based attacks in VANET

The Vehicular Adhoc Network (VANET) consists of connected vehicles with the help of infrastructural modules and Dedicated Short-Range Communication (DSRC). The infrastructural module consists of vehicles, onboard units, RSUs, pedestrians, charging points, and electric grids. The progress of vehicular technology has connected intelligent vehicles and Electric Vehicles (EVs) to the mainframe VANET architecture. As mentioned earlier malware is the prime attack vector, where malware refers to viruses, worms, Trojans, and ransomware. Although malware can launch several types of attacks using different attack dynamics, DDoS is the deadliest among all because of its distribution mechanism and hard detection [7]. Mirai is the most popular and latest malware that uses a botnet-based architecture for DDoS attacks. Other malware capable of launching DDoS attacks include Psyb0t, Chuck Norris, Tsunami, and LUABOT [8]. The general architecture of malware-based DDoS attacks is visible in Fig. 1. In addition to DDoS attacks, there are also the Side Channel Attacks (SCA) in VANETs. A side-channel attack employs indirect measurements related to the vehicle to deduce patterns of usage. This involves monitoring fluctuations in gas levels, changes in the car's weight, engine or cabin heat, tyre wear, paint scratches, and similar factors. Such data points divulge information about the vehicle's usage patterns, travelled distances, etc. These attacks may further help the attacker in launching a DDoS attack by leaking the information. Due to the technological advancements and sustainable growth of ITS, electric vehicles become popular. These electric vehicles are charged through various public charging stations and these charging stations connect to the electric grids. If DoS/DDoS goes undetected, a severe infection can propagate through the VANETs charging points and disrupt the electric grid's services.

1.2. Malware-based SCA in VANET

In the realm of VANETs, malware exploits side channel vulnerabilities to compromise the security and privacy of communications. One avenue through which side channel attacks manifest is traffic analysis, where malicious software scrutinizes the timing and frequency of messages exchanged between vehicles. This enables attackers to deduce sensitive details like routes and destinations. Location-based side channels are another vector, allowing malware to monitor and analyse the broadcasted location updates of vehicles, potentially revealing critical information about their activities. Power consumption analysis becomes a potent side channel in VANETs, where variations in the power usage patterns of in-vehicle devices may expose details about specific cryptographic processes. Leveraging wireless signal strength as a side channel, attackers can gain insights into the proximity and interactions of vehicles. Time synchronization mechanisms within VANETs can also be targeted, enabling malware to manipulate or observe synchronization processes and compromise security protocols. Furthermore, analysing data traffic patterns and monitoring resource utilization in in-vehicle computing systems present additional avenues for side-channel attacks. Protecting VANETs necessitates the implementation of robust security measures, including encryption, secure communication protocols, intrusion detection systems, and privacy-preserving techniques to mitigate the risks associated with malware-initiated side-channel attacks.

1.3. Post quantum cryptography for SCA

Detecting Side-Channel Attacks (SCAs) launched by malware in the VANET involves leveraging advanced cryptographic techniques to fortify systems against unintended information leakage. Post-quantum cryptography focuses on algorithms that remain secure even in VANETs, where traditional cryptographic methods may become vulnerable. By incorporating PQC primitives, such as lattice-based cryptography or hash-based signatures, into a system, it becomes more resilient to the cryptographic vulnerabilities that malware often exploits in SCAs. These advanced cryptographic approaches provide a higher level of security, making it more challenging for malware to glean sensitive information through timing, power consumption, or other side channels. Additionally, the use of quantum-resistant cryptographic techniques helps ensure the long-term security of systems in the face of evolving threats posed by sophisticated malware attempting side-channel attacks. Regularly updating cryptographic protocols and adopting post-quantum cryptographic standards are essential steps in creating a robust defence against malware-initiated side-channel attacks.

1.4. Motivation and contribution

The motivation behind our proposed work lies in addressing the critical need for robust cybersecurity measures in VANETs, particularly in the face of the escalating threat posed by Distributed Denial of Service (DDoS) attacks. As VANETs play a pivotal role in enabling communication and coordination among vehicles, ensuring the integrity and availability of network services is paramount. The emergence of increasingly sophisticated DDoS attacks poses a significant challenge to the reliability of these networks. The envisioned solution, *Protecting vEhicular neTworks against distRibuted deniAl of service attacKs (PETRAK)*, is conceptualized to provide an advanced framework leveraging Machine Learning (ML) techniques, Post-Quantum Cryptography, adaptive thresholds, and multi-layered security protocols. By integrating these cutting-edge technologies, PETRAK aims not only to proactively prevent DDoS attacks but also to detect and mitigate potential threats swiftly, enhancing the overall resilience and security posture of VANETs.

By integrating ML algorithms and leveraging hash-based signatures of post-quantum cryptography, PETRAK endeavours to enhance the security of VANETs. PETRAK efficiently handles malware-based attacks like SCA and DDoS. Malware, such as botnets and Trojan horses, can serve as potent tools in the hands of cyber adversaries to orchestrate large-scale, coordinated DDoS attacks. Malicious software can infiltrate vulnerable systems, establishing a network of compromised devices, often referred to as a botnet. These botnets can then be remotely controlled to flood a target network or service with a deluge of traffic, overwhelming its resources and causing a denial of service for legitimate users. In our simulated scenario, we show a DDoS attack generated through malware. PETRAK has the following contributions:

1.4.1. Multi-layered security

PETRAK offers dual security through attack prevention and detection. The preventive system inspects incoming packets as they enter the network. The detection system examines these packets to guarantee security at a deeper level.

1.4.2. Network-based architecture

We deploy PETRAK at the network level and not in the host devices. Due to this deployment, additional hardware resources are available for attack detection. This deployment aids in early attack detection and can handle more severe attacks. RSUs like signal towers or another reputable entity could be the network location.

1.4.3. Adaptive alarming system

PETRAK uses an Adaptive Alarming Module (AAM) that maintains an adaptive threshold and observes current flow parameters to decide whether there is any abnormality in traffic. This module also handles flash crowds and maintains a separation from attack traffic.

The AAM works on the threshold values decided by the system administrator. The threshold values are crucial for identifying abnormal traffic patterns indicative of potential DDoS attacks. Our approach to threshold calculation is rooted in statistical analysis of traffic parameters obtained during periods of normal, legitimate traffic. The AAM calculates threshold values during these periods, ensuring that they are representative of typical network behaviour. We acknowledge that the initial calculation involves determining maximum and minimum values, but this is only part of a comprehensive statistical analysis. To elaborate, the AAM employs various statistical measures such as mean, standard deviation, and percentiles to establish the normal range of values for each traffic parameter (e.g., vehicle speed, acceleration, density, and inter-vehicle distance). This comprehensive analysis provides a nuanced understanding of the distribution of normal traffic behaviour. Moreover, our adaptive adjustment mechanism is designed to address the dynamic nature of VANET traffic. The thresholds are not static but adapt in real-time based on changing conditions, including factors like time of day, location, weather conditions, road type, and historical data. This adaptivity is crucial for accurately capturing variations in traffic patterns, especially during events like flash crowds, where there can be spikes in legitimate traffic.

1.4.4. Accuracy and speed

PETRAK can detect attacks very accurately with low false positives and very fast. The experimental results show an accuracy rate of more than 99%, which is higher than existing security models. The detection accuracy is 10% higher than the existing solutions. The detection speed determined by model testing is 0.27 s in the generated dataset and is the least among the datasets.

1.5. Paper organization

We have organized the rest of the paper as follows. Section 3 shows the proposed work and defines the functionality of our PETRAK security solution. The outcomes of DDoS mitigation strategies applied in different scenarios are displayed in Section 4. Results for both identification and avoidance from each of the three datasets are displayed individually in this section. Section 5 compares various resulting parameters. This comparison helps in understanding the performance of the proposed solution. We also compare these results with the results of several existing solutions. Finally, Section 6 concludes the results with findings and specifies the limitations of this solution that need resolution in future work.

2. Related work

Before moving towards the proposed PETRAK model, we discuss the existing security methods in further subsections. We divide the security methods into two sections i.e. existing security methods and Enhanced IDS solutions. Existing security methods contain some conventional security techniques whereas subsection Enhanced IDS solutions contain latest security techniques.

2.1. Existing security models

In literature, various resources provide information on VANETs, security attacks, and their solutions. We observe a summarization of DDoS security in [7]. Another summary of DDoS security solutions in vehicular networks is available in [9]. The solutions include identity-based, key-based, trust-based, machine learning, hybrid solutions, and solutions for electric vehicles.

In addition to the above works, we review some recent research in the direction of DDoS security in vehicular networks. Zang et al. (2021) propose a Machine Learning (ML)-based IDS for big data analytics in VANETs [10]. The proposed IDS uses streaming engines to analyse, handle, and visualize massive data. The proposed model uses the Mininet-Wifi environment that simulates the VANET topology connecting all the nodes with mobility. It uses the sFlow technology to gather real-time data and forwards it to the suggested IDS framework. It uses the random forest as the classifier to train and identify abnormal flows. Another Random forest approach for VANET security is noteworthy [11]. A hybrid data-driven methodology for intrusion detection is developed to detect known network breaches and find possible new attackers. It takes advantage of significant data clustering and core sets. This method experiments using a novel dataset, i.e., the CICIDS2017 IDS dataset. A similar solution has been proposed by Goncalves et al. (2021). This study uses ML to provide an intelligent hierarchical security framework for VANET [12]. The proposed system improves threat detection while ensuring robust authentication, privacy, and anonymity. For the experimentation, the proposed model uses a VANET dataset. The results of the experiments shown in the work are easily repeatable and verifiable. The results demonstrate that utilizing various algorithms at hierarchical levels may improve the detection framework. Soni et al. (2022) show a prevention system that protects the communication system from worm-hole and black-hole assaults in VANETs [13]. This system employs the swarm optimization IPS algorithm to identify the malicious vehicle. Particle Swarm Optimization (PSO) uses traffic and verifies the attacker's existence. After leaving the RSU coverage region, the suggested IPS can continue the prevention. Malik et al. (2022) propose a method of blackhole attack mitigation (detection and prevention) in VANET [14]. The solution focuses on identifying blackhole at an early stage of the route discovery process.

In the ever-evolving landscape of cybersecurity, adversaries often employ sophisticated techniques, including the possibility of mounting combined multiple attacks to circumvent traditional defences. An example of such a multifaceted approach is the combination of Differential Power Analysis (DPA) and Differential Fault Analysis (DFA). DPA focuses on exploiting variations in power consumption during cryptographic operations, while DFA targets vulnerabilities induced by injecting faults into a system. The simultaneous execution of DPA and DFA can pose a formidable threat, as attackers leverage both informational leakage and fault-induced vulnerabilities. To thwart such combined attacks, countermeasures must be devised that address the unique challenges posed by each technique. Implementing secure and diverse cryptographic algorithms, incorporating physical security measures to mitigate power side-channel leaks, and ensuring the resilience of systems to injected faults are crucial components of a comprehensive defence strategy. Furthermore, continuous research is essential to stay ahead of evolving attack methodologies, developing countermeasures that can collectively withstand complex and multifaceted adversarial efforts.

2.2. Enhanced IDS solutions

The emergence of quantum computing has sparked significant concerns regarding the security of traditional cryptographic algorithms as well as intrusion detection systems [15]. With the potential for quantum computers to break used encryption methods, researchers are actively exploring approaches. One promising avenue is cryptography, which aims to develop cryptographic tools suitable, for devices with limited resources like IoT sensors and embedded systems. Safeguarding networked systems heavily relies on Intrusion Detection Systems (IDS) which monitor for activities and potential threats. However, the rise of quantum computing poses a challenge to existing IDS as they rely on mechanisms that may become vulnerable. To address this issue researchers are investigating the integration of quantum cryptography with IDS to ensure ongoing protection in the era of quantum technology [16,17]. Another crucial aspect of this discussion is side-channel analysis. Side channel attacks exploit information leakage from an implementation, such as power consumption or electromagnetic radiation. Lightweight cryptography often prioritizes resistance against side-channel attacks making it a vital component in the realm of devices and embedded systems [18,19]. The incorporation of side-channel resistance, into quantum cryptographic algorithms enhances the overall security posture of intrusion detection systems. Ongoing research is being conducted in this area, where different ideas and advancements are being explored to develop a security framework. The goal is to integrate quantum cryptography, lightweight cryptography, and effective side-channel analysis techniques. These efforts are crucial to ensure the strength of intrusion detection systems, in an evolving threat landscape with the emergence of quantum computing capabilities. In the realm of DDoS detection, the implementation of cryptographic algorithms such as Curve448 and Ed448 on resource-constrained processors like the Cortex-M4 is a critical consideration [20]. These elliptic curve components, with Curve448 emphasizing a balance between security and performance and Ed448 tailored for the Edwards-curve Digital Signature Algorithm (EdDSA), pose challenges in optimization for platforms with limited memory and processing power. Similarly, the integration of Supersingular Isogeny Key Encapsulation (SIKE) and its Round 3 variant on Cortex-M4 processors, as well as the implementation of Kyber on 64-Bit ARM Cortex-A processors, demands careful consideration of resource constraints [21,22]. The effectiveness of DDoS detection systems can be significantly impacted by the optimization of cryptographic algorithms on these platforms, ensuring efficient handling of security operations in real time. Additionally, the utilization of cryptographic accelerators on Ed25519 and the exploration of Supersingular Isogeny Diffie–Hellman key exchange on 64-bit ARM architecture contribute to enhancing the security infrastructure, which is pivotal in safeguarding against DDoS attacks. The adoption of these cryptographic techniques not only fortifies communication channels but also reinforces the overall resilience of DDoS detection systems against sophisticated cyber threats [23,24].

In the evolving landscape of cryptographic standards, the NIST lightweight standardization, finalized in February 2023, has become a pivotal reference point. Within the realm of side-channel attacks, fault attacks, a subclass of side-channel attacks, play a significant role in cryptographic vulnerability analysis. Exploring specific implementations, error detection in lightweight Welch–Gong (WG)-oriented stream cipher WAGE has garnered attention. This involves investigating techniques to identify and mitigate errors in the WAGE stream cipher, aligning with the lightweight cryptography paradigm [25]. Similarly, the exploration of error detection in reliable architectures of the Camellia block cipher addresses potential vulnerabilities in this widely-used symmetric key algorithm [26]. Additionally, fault diagnosis in the context of the low-energy Midori cipher is crucial for ensuring the resilience of this lightweight cryptographic primitive against fault attacks [27]. The integration of error detection mechanisms in the block cipher QARMA adds an extra layer of security, addressing potential vulnerabilities arising from faults. These advancements in error detection and fault resilience contribute to the broader goal of fortifying cryptographic systems against sophisticated attacks, including those employed in DDoS detection scenarios [28].

3. Proposed model: PETRAK

We can divide the overall functionality of PETRAK into four main sections or modules. The first module is the Adaptive Alarming Module (AAM), which calculates the threshold and comparative values at regular intervals. The AAM compares these values, and when comparative values exceed threshold values, the AAM generates an alarm by sending an alert message to the system administrator. The AAM uses the Training Module (TM), and the TM further uses the Detection Module (DM). The DM applies the BayesNet ML algorithm to classify the packets and divert the traffic according to the packets' nature. The DM also updates the packet information in the corresponding log files and calls the Prevention Module (PM). The PM uses the logit model for attack prevention. The PM updates the information in the log of malicious packets. In the future, the PM compares the signature of newly arriving packets with the signature stored in the log file or database. There are special scenarios for handling packets.

3.1. Log of benign packets with a higher threshold

The AAM incorporates a log of benign packets that exceed the threshold value but are not considered malicious. This log will help prevent the generation of unnecessary alarms for traffic that falls above the threshold but is harmless. The AAM will maintain a record of these benign packets in a dedicated log file. When comparative values exceed threshold values, the AAM will first check this log of benign packets. If the packet is found in the log, indicating that it is a known benign packet, the AAM will refrain from generating an alarm. This enhancement ensures that the AAM's alerts are specifically focused on potentially harmful activities, enhancing the accuracy of the alarm system and reducing false positives.

3.2. Log of malicious packets below threshold

There may be cases of malicious incoming packets that fall below the threshold. By employing ML algorithms and behavioural analysis, the PM identifies patterns indicative of malicious intent, irrespective of the traffic volume. When the PM observes such patterns, it classifies these packets as potentially harmful and diverts them for further inspection and analysis. Additionally, the PM updates its comparison log files in real-time, ensuring that new patterns and signatures are incorporated promptly. By addressing malicious packets that fall below the threshold, this proactive approach strengthens the system's ability to detect subtle and sophisticated attacks, bolstering the overall security posture of PETRAK. The overall architecture and flow diagram of PETRAK is visible in Fig. 2.

3.3. Packet filtering using hash-based signatures

PETRAK uses hash-based signatures that enhance the security of incoming traffic packets and facilitates effective packet classification in logs. We utilize hash-based signatures as a form of cryptographic verification to ensure the integrity and authenticity of the data. When PETRAK receives a packet, it applies a hash function to its content, generating a unique hash value. This hash value is then encrypted using a private key to create a digital signature. At the receiving end, the digital signatures are decrypted using the corresponding public key, and the hash function is applied again to verify if the computed hash matches the decrypted signature. If they align, it verifies the packet's integrity and confirms that it is indeed sent by the expected source.

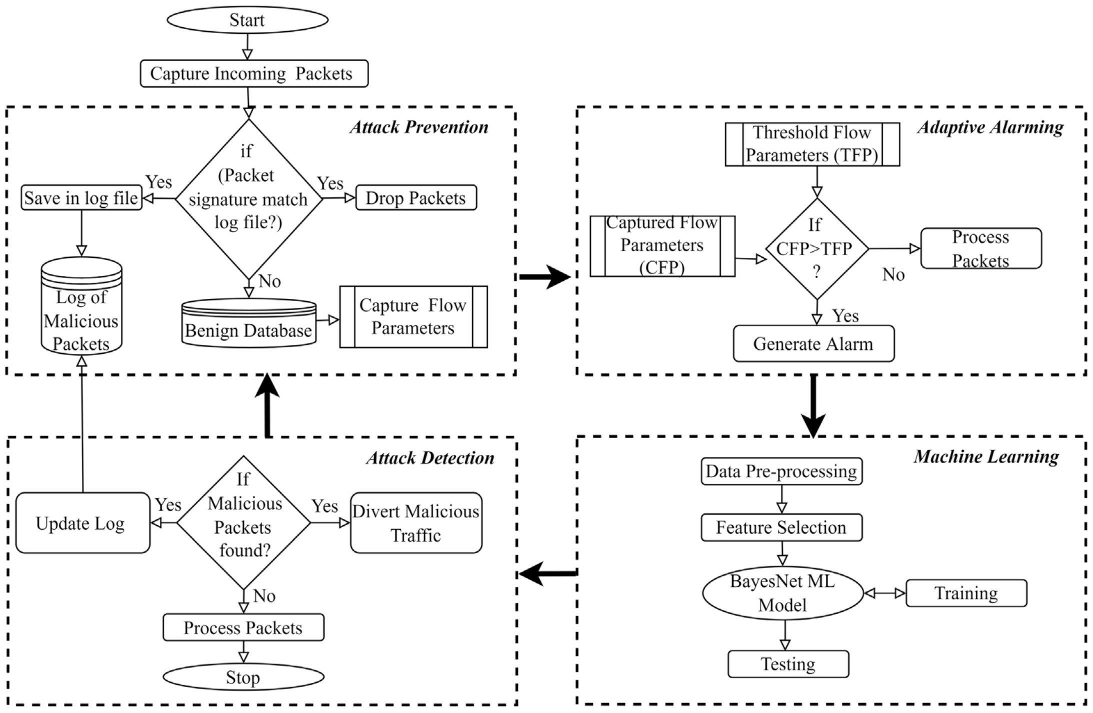

Fig. 2. Flowchart for the proposed PETRAK.

Similarly, hash-based signatures provide an efficient means to categorize and identify packets based on their unique content fingerprints. Each packet's hash value serves as a distinctive identifier, enabling quick and accurate classification within log entries. This helps PETRAK in monitoring and analysing incoming traffic, identifying patterns, and swiftly categorizing packets into predefined classes for further action or analysis. By incorporating PQC-based hash signatures we strengthen the security of our network infrastructure and streamline the management of network traffic. This helps in preventing side channel and DDoS attacks.

3.4. Dataset generation

We use four separate datasets to test the efficacy of PETRAK. These datasets include the generated dataset, the KDDCUP '99 dataset, and the CIC-MalMem-2022 dataset [29,30] and ToN-IoT. We derive the first dataset from the simulation in the NS3 application. The generated dataset is distinctive because it is from a vehicular environment. In this study, our dataset generation process plays a pivotal role in simulating DDoS attacks within vehicular networks. We leverage either publicly available datasets or collect traffic data directly from VANET, tailoring our choices based on the specific problem and model requirements. To ensure relevance and suitability for attack detection, we introduce a synthesized dataset rooted in the attack scenario. This scenario dictates the characteristics of our generated dataset, containing a substantial number of features deemed pertinent for effective attack detection. The process of dataset generation is as follows:

3.4.1. System configuration

The main operating system in use is Windows 11, and for simulation purposes, Ubuntu 20.04.2 LTS is utilized as the virtual operating system. Key software tools incorporated into the system are Network Simulator 3 (NS3), Weka, Python, and Wireshark. The hardware setup includes a dedicated 4 GB graphics card, an Intel i7 13th generation processor, 16 GB RAM, 512 GB SSD, and a 1 TB HDD, ensuring ample computational power and storage capacity for realistic simulation tasks.

Table 1
Attributes of topology.

Parameter	Value
Simulation Platform	NS3.2.7
No. of Vehicles	12
Attacker Nodes	1
Bot Nodes	10
Victim	1
No. of RSUs	1
Routing Protocol	UDP, TCP, ICMP
Visualization Tool	NetAnim
DDoS Rate	20 480 kbps
Normal data rate	512 kbps
Maximum Bulk Bytes	100 000 kbps
Simulation Time	40 s
Data Transmission Rate	100 Mbps
Communication Range	100 m × 100 m
Mobility Model	Random mobility

3.4.2. Topology description

The simulation incorporated a unique attack structure comprising 12 vehicles and a single RSU for communication purposes. Detailed elements include the assigned roles for one attacker, one victim, and 10 bot nodes among the total 12 vehicles. In the communication process, PETRAK uses routing protocols like UDP, TCP, and ICMP. Various parameters like data rates for both normal and DDoS traffic, simulation duration, communication area specifications, and the adoption of a random mobility model for all nodes is as shown in Table 1.

3.4.3. Attack approach

Within the context of the attack scenario, we implement two concurrent types of attacks. Firstly, there is a vehicle-to-infrastructure (V2I) attack, affecting the infrastructure as the RSU is occupied with handling malicious messages. Secondly, a vehicle-to-vehicle (V2V) attack is employed, where a malicious bot node directs a substantial volume of traffic towards the victim vehicle, depleting the victim node's resources in the process of handling this extensive traffic. The attacker strategically utilizes a topology with short distances to enhance the

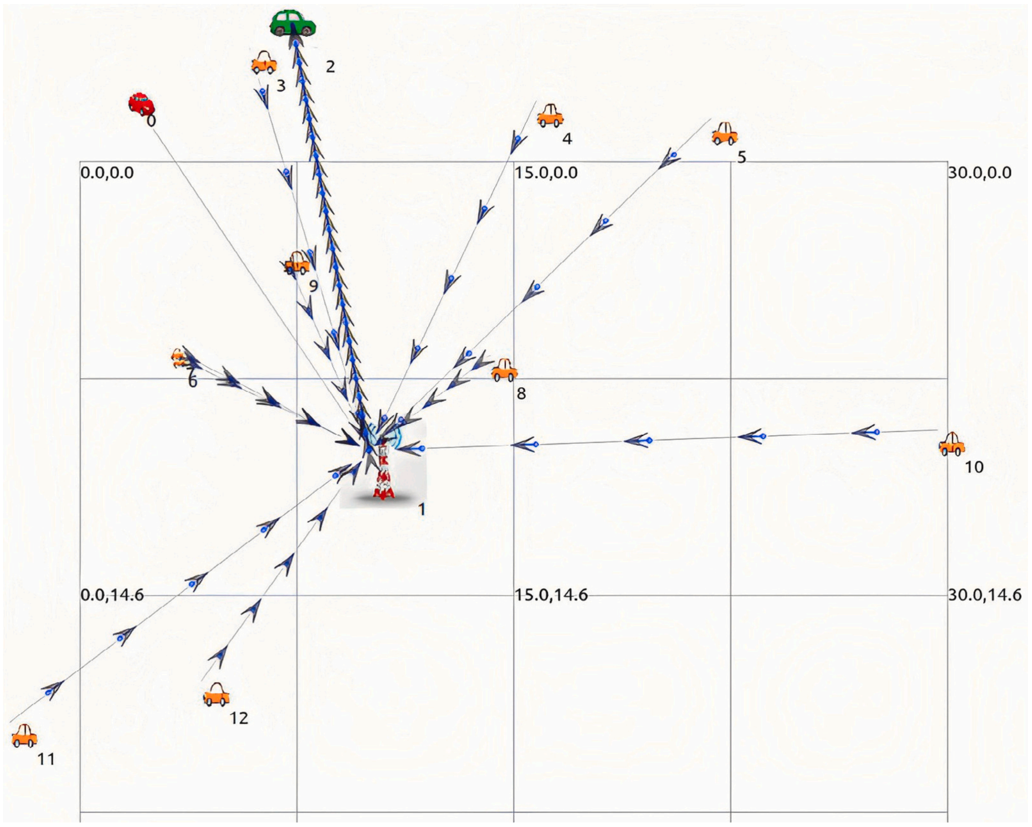

Fig. 3. Vehicle set up in the network.

effectiveness of the attack. This study utilizes a flooding attack (UDP and TCP) generated through a botnet with a peer-to-peer architecture. In this attack strategy, the attacker employs indirect communication methods and does not directly target the victim. The attacker injects malicious code into vehicles, compromising their security. Once compromised, these nodes establish communication with each other, collectively forming a botnet that operates without mutual knowledge, controlled by the attacker. The attack commences as the attacker sends commands to the bot nodes, prompting them to transmit large volumes of data towards the target through the RSU. From an attacker's point of view, this architecture is more reliable because it is challenging to close an attack from this architecture. Multiple layers of communication easily hide the mastermind during the attack. In this paper, we have tried to detect this kind of attack with various layers of security.

This paper endeavours to detect such attacks by implementing various layers of security. Three distinct scenarios are simulated to encompass packet features of all types. *1. Normal Scenario:* Vehicles communicate without any attack, with only benign traffic present. *2. Attack-only Scenario:* This scenario focuses on a flooding attack without any benign traffic, assessing the system's capacity and performance under a substantial influx of incoming traffic. *3. Mix Scenario:* This scenario involves a DDoS attack alongside benign traffic, encompassing UDP and TCP floods as two types of flooding attacks. The attack process is shown through the following Figure 3.

3.4.4. Data capturing and dataset generation

The dataset creation involves packet tracing and movement monitoring in NS3, where we focus on the target node using the 'pcap' option. The resulting 'pcap' file is processed in Wireshark, selecting and saving relevant packet attributes as a .csv file. Our synthetic dataset comprises 28 attributes and 697,792 records, embodying the intricacies of the attack scenario. It is essential to note that the dataset reflects the nuances and patterns inherent in DDoS attacks within vehicular networks, distinguishing it from existing datasets. This distinction in attack methodology and the resultant dataset characteristics are critical factors that contribute to the uniqueness and relevance of our research in the domain of vehicular network security. Unlike some existing datasets that may lack specificity to the vehicular network context, our synthesized dataset is crafted with a focus on the DDoS attack scenario. This scenario imparts unique characteristics to our dataset, capturing the intricacies and patterns associated with DDoS attacks in the context of vehicular communication. The attributes within our dataset are carefully selected to align with the nuances of the attack methodology, ensuring its relevance to the challenges posed by security threats in vehicular networks.

3.5. Data pre-processing

For data pre-processing, we delete the duplicate records, replace missing and null values, and change the data types. Missing and null values are replaced by 0 or some other appropriate value. Then we adjust the class labels, keeping only two classes to categorize the occurrences using a binary classification technique. We divide each dataset into two independent datasets: one for training and one for testing. The split ratio is 70:30, where 70 is for training, and 30 is for testing. Before the data is split, all the instances and records are randomized to break the sequence of malicious and benign packets in the dataset.

3.6. Feature selection

For feature selection, the DM uses a supervised learning method *CFS Subset Evaluator* with the best first search method in WEKA. This evaluator determines the most relevant attributes for the classification. Sometimes this evaluator selects fewer attributes than specified in the subset evaluator properties. In PETRAK, the evaluator proposes different features for each dataset. The PM also uses feature selection in the STATA application using Wald's z-test. The PM selects only that feature set that gives probability values under the specified limit.

3.7. Implementation

The Proposed PETRAK model works in different phases by dividing the complete process into various small functions. These functions are prevention, alarm, training, and detection.

3.7.1. Prevention function

PETRAK prevents attacks or intrusions in two ways. The first method prevents an attack immediately, and the other is suitable for future prevention. PETRAK uses a binary logit regression-based solution for immediate attack prevention that separates malicious packets from legitimate packets. It is a statistical tool based on probabilities to determine their class. The approach also uses Wald's Z-test, LR Chi (likelihood-ratio statistics), and Hosmer–Lemeshow goodness of fit tests. For future attack prevention, the PM stores the signature information of malicious packets and malware in a separate database. The PM uses this history-based database to compare the signatures of all incoming packets. If incoming packet signatures match the database, the PM drops those packets, labelling them malicious.

3.7.2. Alarming function

PETRAK has an Alarming Module (AM) that uses flow parameters of the communication channel to raise alerts and start detection. The AM uses flow parameters like delay, jitter, sent bitrate, flow IDs, etc., and calculates their values. The AM compares these values with some pre-defined threshold values. The AM triggers an alarm if it notices parameter values that are higher than the threshold values. Additionally, the AM uses hardware resource consumption metrics like CPU and memory usage. Excessive consumption of these hardware resources and higher values of flow parameters like delay and jitter, as well as the number of flow IDs, indicate suspicious activity in the network. We find the highest traffic value observed during normal traffic and use it as the threshold. CalculateThreshold Process includes the following steps:

Data Collection: Collect historical data for network traffic volume in megabytes (MB) during different time intervals. For this example, let us consider data for a specific network segment:

Traffic Data (MB) for Morning Hours: [250, 260, 270, 280, 290, 300, 310, 320, 330, 340] Traffic Data (MB) for Evening Hours: [280, 290, 300, 310, 320, 330, 340, 350, 360, 370] Threshold Calculation:

To calculate the upper threshold, we simply find the maximum traffic value observed during normal traffic hours.

$$UpperThreshold = max(TrafficData(MB)) \tag{1}$$

Upper Threshold for Morning Hours = Maximum value in the morning data = 340 MB Upper Threshold for Evening Hours = Maximum value in the evening data = 370 MB.

The rationale behind the selection of threshold values in the proposed approach is critical for effectively handling Distributed Denial of Service (DDoS) attacks. Opting for higher threshold values is rooted in the understanding that DDoS attacks typically involve a substantial influx of malicious traffic aimed at overwhelming the targeted system. By setting higher thresholds, the system becomes more resilient to variations in regular traffic patterns and can better differentiate between normal fluctuations and abnormal spikes indicative of an ongoing attack. Higher thresholds enable the system to accommodate the dynamic nature of network traffic, providing a buffer that allows legitimate traffic to pass through unhindered while raising an alert or triggering mitigation measures when the traffic surpasses the predefined threshold. This approach is technically justified as it enhances the sensitivity of the detection system, making it more adept at discerning and responding to anomalous patterns associated with DDoS attacks, ultimately bolstering the network's ability to withstand malicious attempts to disrupt service availability.

3.7.3. Dynamic adjustment

In the context of VANET, employing dynamic adjustment in thresholds is crucial for accurately responding to fluctuating demands and potential anomalies. By continuously monitoring traffic patterns and adapting thresholds in real-time based on current network conditions, the system can effectively differentiate between regular traffic fluctuations and abnormal activities such as DDoS attacks. Dynamic adjustments allow thresholds to be recalibrated promptly, ensuring that they remain relevant and sensitive to changing demands, preventing false alarms during periods of high legitimate traffic, and enabling timely detection of suspicious or disruptive network behaviour. This adaptive approach enhances the network's ability to respond dynamically to varying loads and emerging threats, ensuring optimal performance and security.

Traffic rates remain different at different periods. For example, during night hours traffic volumes are low while during the day time traffic remains high. If we keep the same threshold for all the periods then either we will face false alarms or DDoS attacks may become undetectable. If we keep a low threshold during daytime then we will face lots of false alarms as usually there remains high traffic during day time. Similarly, if we keep a high threshold during the night then the system will not generate any alarm during a DDoS attack considering it as normal traffic. That is the reason behind the dynamic adjustment of thresholds.

Anomaly Detection: Whenever the current traffic volume exceeds the upper threshold, it triggers an alert or takes appropriate action. For example, if the network traffic exceeds 370 MB during the evening hours, it would trigger an alert. In this example, we determine the upper threshold for network traffic by selecting the highest observed value during normal traffic conditions for both morning and evening hours. This approach ensures that the threshold is set to accommodate the highest expected traffic load and allows for the detection of unusually high traffic volumes.

After observing any suspicious activity, the AM sends an alarming signal to the detection module. The AM's threshold values continuously change at different time intervals, which is a crucial characteristic. Because traffic volumes fluctuate depending on the time of day, threshold values should be flexible.

3.7.4. Detection function

The DM's machine learning model starts working once the AM issues an alarm. For classifying packets, the DM employs the BayesNet machine learning algorithm. Malicious and malware-based packets are distinguished from benign packets by the DM. To implement detection, the DM employs the WEKA application and uses a trained ML model to identify the attack packets quickly and reliably. The DM reroutes the malicious packets via sinkhole, blackholing, or any other technique. PETRAK maintains a comprehensive understanding of the current threat landscape. This real-time intelligence enrichment enhances PETRAK's ability to recognize sophisticated attack patterns, allowing it to respond effectively to targeted and highly adaptive threats. The implementation process is visible with the help of the following algorithm 1.

The PREVENT procedure begins by taking an attribute x as input and proceeds with a repeat-until loop, iterating until a model is selected. Within this loop, it conducts a Z-test on each attribute in the dataset, computing the Z-score based on mean and standard deviation. A significance level ($p > z$) is calculated using the Z-score and the cumulative distribution function. If this significance level is less than 0.05, indicating statistical significance, the attribute is selected for the model; otherwise, it is rejected. The selected attributes are then used in a logistic regression model. The goodness of fit is assessed using the Chi-squared statistic (X^2), comparing observed and expected values, and the Hosmer–Lemeshow GoF (X^2HL), considering group size and count. The expected values are the probabilities of observing a specific outcome (such as 1 or Yes) for each case in the dataset, as predicted by the logistic regression model. These expected probabilities are then

Algorithm 1 Proposed PETRAK algorithm

```
1: import *.*
2: Input: $TrafficParameters$
3: Output: $ClassifiedPackets$
4: procedure ALARM
5:     Initialize-all-the-parameters-as:
6:     $interval1 \leftarrow time1$;
7:     $interval2 \leftarrow time2$;
8:     $stTimeThr \leftarrow time3$;
9:     $stTimeComp \leftarrow time4$;
10:    Timer-procedure-is-used-to-calculate-threshold-values
11:    procedure TIMER(interval2,stTimeThr)
12:        $thr = calculateThreshold()$;
13:        $return(thr)$;
14:    procedure CALCULATETHRESHOLD
15:        $PythonScript()$;
16:    procedure TIMER(interval, stTimeComp)
17:        $comp = calculateParameter()$;
18:        $return(comp)$;
19:    procedure CALCULATEPARAMETER
20:        $PythonScript()$;
21:    if $comp \geq thr$ then
22:        Raise alarm;
23:        TRAIN();
24:    else
25:        $processpackets$;
26:        $continue()$;
27: procedure TRAIN
28:    Data Pre-processing
29:    Feature & algorithm Selection
30:    Train the ML model
31:    DETECT                              ▷ DETECT() Procedure Call
32: procedure DETECT
33:    Use the dataset
34:    if Packet = Malicious then
35:        Divert Traffic and Update Log Files
36:        PREVENT                         ▷ PREVENT() Procedure Call
37:    else
38:        Process Packet
39: procedure PREVENT
40:    Input: Attribute x
41:    Output: Accept or reject decision
42:    $z = (x - x_{mean})/std.dev$
43:    $(p > z) = 2 * (1 - pnorm(z))$
44:    if $((p > z) < 0.05)$ then
45:        select attribute
46:    else
47:        reject attribute
48:    Apply logistic model
49:    $X^2 = \sum \frac{(O_i - E_i)^2}{E_i}$
50:    $X^2HL = \sum_{g=1}^{G} \frac{(O_g - E_g)^2}{E_g(1 - E_g/N_g)}$
51:    compute $p$
52:    if $(MV > SV)$ then
53:        select model
54:    else
55:        reject model
```

Table 2
Flow parameters in different simulation scenarios.

Parameters	Normal	Mixed traffic	UDP flood
Average Packet Loss Ratio	0	30.873	46.096
Average Sent bitrate	0.315	12 001.655	18 006.164
Average Received bitrate	0.315	12 001.655	18 006.164
Average Mean delay	1.027	27.041	35.588
Jitter in milli/seconds	0	541.26	415.620
Flow IDs Generated	5	12	18

used to compute the Chi-squared statistic (X^2), which measures the difference between the observed outcomes and the outcomes expected by the model. The model's variance and structural variance are compared, and if the model variance (MV) surpasses structural variance (SV), the model is selected; otherwise, it is rejected. The chosen model's data is exported for use in machine learning applications. The dataset is split into a 70–30 ratio for training and testing, respectively. LogitBoost with Decision Stump is initialized and trained with the training features. Predictions are made for each observation in the testing dataset, and the predicted labels are stored. Finally, the algorithm evaluates the model's performance metrics, including accuracy and precision, providing a comprehensive framework for feature selection, model selection, and predictive modelling.

4. Results and discussion

We initiate this section with a discussion of an adaptive alarming module and its results. Further, we discuss the results of detection for each dataset used in the experiments followed by the prevention results. At the end, we also provide some advanced implications related to side-channel attacks and post-quantum cryptography.

4.1. Adaptive alarming module

PETRAK uses flow parameters to generate an alarm for the prevention of attacks. The proposed solution captures network parameters and raises alerts for the detection system. Various parameters for generating these alarms are the average sent bit rate, the average received bit rate, the average packet loss ratio, and the average mean delay, jitter, and flow IDs. A sudden rise in these parameters suggests a possible DDoS attack. In the case of a flash crowd, threshold values change accordingly, and an alarm works when computed flow parameter values exceed threshold values. Table 2 shows the increase in parameter values.

From Table 2, it is clear that parameter values increased with the launch of a DDoS attack. Other observations have emerged from the results like an increase in the number of flow IDs in attack scenarios. Packet Loss Ratio increased significantly from 0% to 46%, indicating abnormal behaviour. Average Mean Delay also increased from 1 ms to 35 ms, which is abnormal behaviour. The Average Sent bitrate also increased significantly from almost .5 kbit/s to 18007 kbits/s. The jitter increased from 0 to 541.26 in the mixed scenario and 415.62 in the UDP flood attack.

4.2. Detection

It is the procedure that determines if an assault has occurred or not. Prompt assault detection by the DM always results in the least amount of damage to the target and the quickest recovery. Delays in assault identification lead to lots of problems like escalation lateral movement and extended downtime.

4.2.1. Generated dataset

The dataset contains records of different *.pcap files*. These pcap files contain 697792 records and 27 attributes. We use the CFS Subset Evaluator algorithm with the Best First Search (BFS) method for feature selection.

In the detection, we use the BayesNet classifier with the K2 searching algorithm. This search algorithm uses a random order of variables as input and applies scores during its functionality. The performance of the algorithm is heavily affected by the sequence of the parameters. Wrong parameter ordering may result in an erroneous learning network structure. After getting a network structure, we select a class in the estimate package to determine how to learn the probability tables. We use the Simple Estimator Class (SEC), which estimates conditional probabilities directly using the following Eq. (2).

$$P(x_i = k|pa(x_i) = j) = \frac{N_{ijk} + N'_{ijk}}{N_{ij} + N'_{ij}}. \tag{2}$$

In Eq. (2), N'_{ijk}, i.e., the alpha parameter is set at the default value of 0.5. The detection model applies filters like replace-missing-values, string-to-nominal, and remove-percentage-split. The detection process uses a total of 209338 instances for testing and three attributes, i.e., time to live, epoch time, and class. The model's building and testing on the provided test set each took 0.03 and 0.27 s, respectively.

4.2.2. KDDCUP'99

We use a compact version of KDDCUP '99 to avoid heavy computations, which is only 10% of the original dataset. Although this dataset contains fewer instances, these are good for ML training. The 10% KDDCUP'99 contains all necessary attributes, such as basic, traffic, and content features, despite removing the most redundant records. We use the CFS Subset Evaluator algorithm to select features with the best first search method. The evaluator selects five attributes out of 41 attributes and one class attribute. The properties like dst_host_count, logged_in, sub_class, dst_bytes, and srv_diff_host_rate, etc. make up this list. We use 70% of the 494021 illustrations in the dataset in the training phase, while testing uses the remaining 30%. We put the Bayes Network Classifier with InputMappedClassifier to the test on 148206 instances. BayesNet uses the k2 searching algorithm for training the model and a simple estimator class that estimates conditional probabilities during the testing phase. The model's creation took 98.83 s, while its testing on the available test set required 0.77 s. We obtain a classification accuracy of 99.9% while the precision, recall, and F-measure yield the same values.

4.2.3. CIC-MalMem-2022

CIC-MalMem-2022 contains signatures of obfuscated malware that remain hidden but keep harming the machine and escape detection. This dataset contains 58596 instances, of which 29298 are malicious, and 29298 are benign. We use the CFS subset evaluator algorithm for feature selection with the best first search method. We select only the five most relevant attributes. We split the 58596 instances into $70\% - 30\%$ ratios for training and testing. We use the Bayes Network Classifier with InputMappedClassifier on 41017 records for training and 17579 instances for testing. To maintain uniformity, we apply the BayesNet classifier in detection. The BayesNet uses the k2 searching algorithm for training the model and a simple estimator class that estimates conditional probabilities during the testing phase. The model took 98.83 s to construct, and 0.77 s to validate the model using the provided testing set. During the testing phase, the PETRAK model achieves a classification accuracy of 99.69%, and the precision, recall, and F-Measure parameters yield the same results at 99.7%. Table 3 shows detailed results with different parameters.

4.2.4. ToN-IoT

The ToN-IoT dataset is also utilized in the present research. Various sources like the IIoT system's network traffic, Windows operating system logs, Linux operating system logs, and telemetry data from linked devices are a few of the data streams that make up ToN-IoT. Diverse data has been provided through a mid-sized IoT network. ToN-IoT has been developed by UNSW Canberra IoT Labs and The Cyber Range. The network ToN-IoT dataset is available in the ToN-IoT repository [8]. Additionally, ToN-IoT statistics are shown in a CSV file with a marked column designating whether an assault or regular operations are taking place. Attack types such as data injection, backdoors, scanning, DoS, and DDoS are included under the subcategory "attack type" in the MITM. These attacks against IoT and IIoT sensors were launched and gathered over the IIoT network.

4.2.5. CICEV2023

The DDoS Attack Dataset (CICEV2023) is a specialized and essential resource designed to address a critical gap in cybersecurity research related to Electric Vehicle (EV) charging infrastructure. With a focus on DDoS attacks against EV authentication, CICEV2023 stands out as a unique dataset providing diverse machine-learning features. It goes beyond conventional datasets by incorporating not only the reception count of packets during specific periods but also includes valuable information such as packet access counts and system status details on charging facilities. The dataset introduces four distinct attack scenarios, namely Correct EV ID, Wrong EV ID, Wrong EV Timestamp, and Wrong CS Timestamp, offering a comprehensive exploration of potential security threats in the realm of EV charging systems. CICEV2023 plays a crucial role in contributing to the analysis of EV charging systems and serves as a valuable resource for training and testing DDoS attack detection classifiers, fostering advancements in the understanding and mitigation of cybersecurity risks in the electric vehicle domain.

4.2.6. VeReMi

The Vehicular Reference Misbehaviour Dataset (VeReMi) is a pivotal contribution to the field of vehicular network security, providing a simulated dataset to assess misbehaviour detection mechanisms. Generated using LuST and VEINS, VeReMi serves as a benchmark for evaluating the effectiveness of security measures in large-scale urban environments. This dataset encompasses message logs per vehicle, incorporating GPS data and Basic Safety Message (BSM) messages exchanged through Dedicated Short-Range Communication (DSRC). VeReMi offers versatility with three density levels, five unique attacks, and three attacker density variations. By offering transparency in its generation process, VeReMi enables researchers to reproduce and extend the dataset for further investigations. The dataset's extensibility encourages the exploration of new attacks, changes in simulation parameters, and diverse scenarios. VeReMi significantly aids the development and evaluation of misbehaviour detection mechanisms in vehicular networks, fostering advancements in the cybersecurity domain.

The following Table 3 shows various parameters like precision, recall, and many more.

We use the abbreviations of Cl1 as True Positive Rate, Cl2 as False positive Rate, Cl3 as Precision, Cl4 as Recall, Cl5 as F1-score, Cl6 as MCC, Cl7 as ROC area, and Cl8 as PRC area in Table 3. Packet label "M" refers to malicious packets, whereas "B" refers to benign packets. Abbreviations for the ToN-IoT dataset are N for Normal, B for Backdoor, D for DDoS, I for Injection, P for the password, R for Ransomware, S for Scanning, and X for XSS.

According to the formulated dataset's specifications and KDDCUP'99, the TPR is 99.9%, the FPR is 0.00%, and the F1-score is 99.9%. The CIC-MalMem-2022 dataset's features indicate TPR of 99.7%, FPRof 0.003%, and F1-score of 99.97%. Overall, the findings indicate that PETRAK is effective in detecting malware in vehicular networks.

Table 3
Detailed correctness by class in various datasets.

Cl1	Cl2	Cl3	Cl4	Cl5	Cl6	Cl7	Cl8	Class
GENERATED DATASET								
0.999	0.000	1.000	0.999	1.000	0.998	1.000	1.000	M
1.000	0.001	0.997	1.000	0.998	0.998	1.000	1.000	B
Weighted Average								
0.999	0.000	0.999	0.999	0.999	0.998	1.000	1.000	
KDDCUP'99 DATASET								
0.999	0.000	1.000	0.999	1.000	0.998	1.000	1.000	M
1.000	0.001	0.997	1.000	0.998	0.998	1.000	1.000	B
Weighted Average								
0.999	0.000	0.999	0.999	0.999	0.998	1.000	1.000	
CIC-MALMEM-2022 DATASET								
0.995	0.001	0.999	0.995	0.997	0.994	1.000	1.000	B
0.999	0.005	0.995	0.999	0.997	0.994	1.000	1.000	M
Weighted Average								
0.997	0.003	0.997	0.997	0.997	0.994	1.000	1.000	
ToN-IoT DATASET								
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	N
0.921	0.000	1.000	0.921	0.959	0.957	1.000	0.998	B
0.588	0.000	0.971	0.588	0.732	0.752	0.996	0.857	D
0.989	0.006	0.648	0.989	0.783	0.798	0.999	0.892	I
0.996	0.001	0.977	0.996	0.986	0.986	1.000	0.992	P
0.998	0.005	0.478	0.998	0.646	0.689	0.998	0.704	R
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	S
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	X
Weighted Average								
0.988	0.000	0.992	0.988	0.988	0.988	1.000	0.994	
CICEV2023 DATASET								
0.964	0.000	0.992	0.991	1.000	0.992	0.991	0.991	M
0.993	0.001	0.991	0.993	0.998	0.992	0.993	1.000	B
VeReMi DATASET								
0.991	0.991	0.992	0.991	0.991	0.990	0.990	0.990	M
0.965	0.035	0.991	0.991	0.991	0.991	0.991	0.991	B

Table 4
Logit model results of generated dataset.

Class label	Co. eff.	Stnd-Error	Z	P >Z	95% Confi. interval	
L1	0.000	0.000	4.160	0.000	0.000	0.000
L2	1.634	1.073	1.520	0.128	−0.468	3.738
L3	0.009	0.001	4.720	0.000	0.005	0.012
L4	0.002	0.001	1.890	0.059	0.000	0.004
_cons	−1.313	0.740	−1.770	0.076	−2.765	0.138

Table 5
Logit model results of KDDCUP'99 dataset.

Class label	Co. eff.	Stnd. Error	Z	P >Z	95% Confi. interval	
L1	−9.079	0.282	−32.17	0.000	−9.632	−8.526
L2	10.842	0.281	38.500	0.000	10.290	11.394
L3	−0.010	0.000	−84.69	0.000	−0.010	−0.010
L4	5.437	0.122	44.390	0.000	5.197	5.677
L5	0.027	0.000	97.67	0.000	0.026	0.027
_cons	−1.474	0.020	−72.11	0.000	−1.514	−1.434

Table 6
Results of logit model from CIC-MalMem-2022 dataset.

Label	Co-eff	Std-Err.	Z	P >Z	95% Conf. interval	
L1	−0.178	0.031	−5.700	0.000	−.240	−.117
L2	−.016	.000	−60.440	0.000	−.016	−.015
L3	0.899	0.022	40.450	0.000	0.855	0.942
L4	0.067	0.013	4.870	0.000	0.040	0.094
L5	−.003	0.016	−0.220	0.829	−0.036	.029
_cons	16.816	0.396	42.44	0.000	16.040	17.593

4.3. Prevention

To prevent attacks, we apply Wald's Z-test to the dataset. We also obtain individual probabilities of attributes to determine which features are suitable for the prevention model. We evaluate the overall efficiency of the model using the Hosmer–Lemeshow Goodness of fit test. We use the probability, or p-value, to see the effectiveness of attributes and models, where 1.00 signifies a perfect fit. We check the individual packet probability using the coefficients to differentiate between malicious and benign packets.

4.3.1. Generated dataset

In this dataset, the number of observations is 697792, with LR chi2(4) = 60.01, Prob >chi2= 0.0000, pseudo R2 = 0.0015, and log-likelihood = −19632.139. PETRAK uses six iterations to converge. We show the other observed parameters in Table 4.

In Table 4, L1, L2, L3, and L4 stand for destination-port, stream index, tcpsegmentlen, and time to live, respectively. This table also displays the Z-test findings, which reveal that none of the $P>z$ values are higher than the 5% significance level. We accept only those variables that have the correct values. For $P >z$, two variables, destination-port, and tcpsegmentlen, have 0 values, whereas stream index and time to live have 0.128 and 0.059, respectively. The fact that none of the variable coefficient values are close to zero means that each one has a statistically significant effect on the dependent variable *class*. The range of both values in the *95% confidence interval* column is neither 0 nor $Near0$, demonstrating the statistical significance of the variables.

4.3.2. KDDCUP'99

In this dataset, we use 494021 observations with LR chi2(5) = 442899.77, Prob >chi2= 0.0000, Pseudo R2 = 0.9036 and Log likelihood = −23628.865. The model took 8 iterations to converge, and other observed parameters are shown in Table 5.

In Table 5, attribute names from L1 to L5 stand for rerror_ratecontinuous, srv_rerror_ratecontinuous, dst_host_srv_countcontinuous, dst_host_serror_ratecontinuous, and count continuous, respectively. The Z-test results are also included in this table, and they demonstrate that all of the $P>z$ results are lower than the 5% significant threshold. We consider the variables with such values in parameter selection. The fact that none of the variable coefficient values are close to zero means that each one has a statistically significant effect on the dependent variable *class*. The range of both values in the *95% confidence interval* column is neither 0 nor $Near0$, demonstrating the statistical significance of the variables.

4.3.3. CIC-MalMem-2022

In this dataset, the number of observations are 58596 with LR chi2(5) = 77377.71, Prob >chi2= 0.0000, Pseudo R2 = 0.9526 and Log likelihood = −1926.7968. The model took 7 iterations to converge. other observed parameters are shown in Table 6.

In Table 6, attribute names from L1 to L5 stand for pslistnppid, handlesnevent, handlesndesktop, malfindninjections, and psxviewnot_in_session, respectively. Results obtained from the Logit model using the CIC-MalMem-2022 dataset show that p values are again less than the significance level for all attributes except psxviewnot_in_session attribute. This attribute has a higher $P>z$ value of 0.829.

4.4. Analysis of PETRAK for advanced implications

Our proposed PETRAK is generalized in nature. Therefore, the PETRAK solution is also applicable for some advanced implications such as the detection of Side Channel Attack (SCA) and the detection of quantum attacks. We provide a theoretical analysis of the detection process of the above-mentioned attacks by our proposed PETRAK. The implementation details will be in future scope subject to the availability of the dataset requirements.

4.4.1. PETRAK for SCA detection

PETRAK's BayesNet is applicable for detecting SCAs by modelling dependencies between variables and assessing the likelihood of observed events. In this context, the algorithm analyzes system variables and their correlations to identify abnormal patterns indicative of side channel activity. By constructing a Bayesian Network that represents relationships between variables, such as power consumption or timing, the algorithm can learn normal behaviour and identify deviations. The detection involves calculating the probability of observed side channel patterns given the learned network structure. Significant deviations from expected behaviour trigger alerts, signalling potential SCAs. Additionally, Bayesian Networks enable dynamic updating as new data is acquired, improving adaptability to evolving attack methods. The approach provides a probabilistic framework for SCA detection, leveraging statistical inference to enhance system security. We summarize a probable process for the above in Algorithm 2.

Algorithm 2 SCA detection approach by proposed PETRAK

```
    Input: $O, B, th$
    Output: $D$
 1: Define $B : build\_bayesian\_network(nodes, d)$
 2: Define $nodes : [Power, Timing, Data]$
 3: Define $d = Power : [Timing, Data], Timing : [Data]$
 4: Define $train\_bayesian\_network(normal_d ata)$
 5: Define $detect\_side\_channel\_attack(O, th)$
 6: Calculate $L : P(O|B)$
 7: if $L > th$ then
 8:     Return $(D = True)$
 9: else
10:     Return $(D = False)$
```

The algorithm inputs observed values of traffic (O), Bayesnet model (B), and a threshold (th); the algorithm outputs the decision of the detection (D) based on the comparison of th and O. B is defined based on nodes and dependencies (d), where nodes relate to power, time, and data; d relates to the power in the form of time and power value (data), time relates to only time taken of operation (data). The mode B is trained on normal data and we can detect anomalies based on the comparison between threshold th and the observed values O.

4.4.2. PETRAK for PQC-based attack detection

Post Quantum Cryptography (PQC) is the pillar of future network security as the PQC provides sustainable resilience in encrypted traffic. However, PQC is vulnerable to various attacks such as Quantum Key Distribution (QKD) attacks, Lattice-based attacks, Code-based Attacks, Multivariate Polynomial Attacks, and Hash-based Attacks. Though our PETRAK has not been experimented directly to detect PQC attacks, we provide a summarization that how BayesNet of our proposed PETRAK can be used to detect such attacks subject to the availability of the datasets. We provide the summary in Table 7. We keep the scope open for future researchers to generate datasets and design ML algorithms for detecting such attacks.

5. Comparative analysis

Within this section, we conduct a comparative analysis of detection and prevention results across various datasets. When evaluating detection parameters, we assess factors such as the time required to construct and test the model, the accuracy rate, precision, recall, and F-measure at different error rates. In the prevention tests, we consider sensitivity and specificity, as well as positive and negative predictive values. Additionally, we compare PETRAK with other models that have emerged in recent years for a comprehensive evaluation.

5.1. Comparison of PETRAK with different datasets

Our observations indicate that detection and prevention techniques exhibit superior performance when applied to generated or synthesized datasets. Specifically, in the detection process, time-related parameters demonstrate better results for the generated dataset. The “Time to Build Model in Seconds” is recorded as 0.3, while the “Time to Test Model in Seconds” is 0.27. Furthermore, the classification accuracy in the generated/synthesized datasets reaches an impressive 99.99%, surpassing the accuracy rates of 99.93% for the KDDCUP'99 dataset and 99.69% for the CIC-Malmem-2022 dataset. The generated/synthesized dataset also exhibits better values for TP Rate, FP Rate, and recall. For a comprehensive overview of the detection technique's results across different datasets, please refer to Table 8.

The obtained results from applying the prevention technique on different datasets are presented in Table 9. When focusing on DDoS prevention, the generated/synthesized dataset demonstrates the highest sensitivity, achieving a value of 100%. In comparison, the sensitivity values are 98.53% for the KDDCUP'99 dataset and 99.35% for the CIC-Malmem-2022 dataset. Furthermore, the classification accuracy in the generated/synthesized dataset attains the highest value of 99.56%, outperforming the accuracy rates of 98.63% for the KDDCUP'99 dataset and 99.17% for the CIC-Malmem-2022 dataset. The generated/synthesized dataset exhibits better values for TP Rate, FP Rate, and recall.

5.2. Comparison of PETRAK with existing models

After comparing PETRAK results for three datasets, in Table 8, we compare our proposed PETRAK with some existing solutions in Table 10. We use parameters like detection type, ML algorithm, and dataset used by the process. We compare all the results based on accuracy in Table 10.

Our proposed framework PETRAK is better among these solutions as it does attack prevention with attack detection. Two separate approaches, i.e., logistic regression and machine learning, are used for prevention and detection tasks. As we test our PETRAK model using different datasets, we claim that PETRAK is a generalized solution for various vehicular networks. Our solution produces similar results in all cases. We also compare these solutions based on accuracy and find that our solution, PETRAK provides the highest accuracy.

6. Conclusion and future work

From the above work, it is evident that our proposed model, PETRAK, is appropriate for malware-based DDoS attacks. In the experiments, all datasets obtain classification accuracy higher than 99%. Apart from the detection, the logistic regression-based prevention helps our proposed model to be an efficient security solution for vehicular networks. However, some parameters in PETRAK could be improved, such as the specificity of attack prevention. In the future, we would like to implement the high-coordinated mapping-based detection and prevention model based on DL methods. PETRAK's future development will include the integration of User Behaviour Analytics (UBA) modules. By analysing user behaviour patterns within the network, PETRAK can differentiate between legitimate user activities and potentially malicious actions. UBA integration will provide a more nuanced understanding of network interactions, enabling PETRAK to identify anomalies related to user behaviour, and further enhancing its threat detection accuracy. Similarly, to meet the demands of modern, large-scale networks, PETRAK will undergo enhancements to ensure scalability and compatibility with cloud-based infrastructures. By optimizing its architecture for cloud deployment, PETRAK can seamlessly adapt to varying network sizes and configurations, making it a robust solution for both enterprise and cloud-based environments.

Table 7
Attacks on PQC and Bayesnet solution approach.

Attack category	Target	Description	Bayesnet detection approach
QKD attacks	Quantum key distribution protocols	Although QKD is considered a quantum-safe method for key exchange, certain practical implementations may be vulnerable to various side-channel attacks or flaws	Train a BayesNet model on features reflecting normal QKD systbehaviour, such as quantum state parameters and communication patterns. Calculate the probability of observed deviations indicative of potential QKD attacks
Lattice-based attacks	Lattice-based cryptographic schemes	Lattice-based constructions may be susceptible to certain attacks like the Short Integer Solution (SIS) or Learning With Errors (LWE) problem	Employ a BayesNet model trained on features related to mathematical operations in lattice-based cryptography. Assess the probability of observed pattersignalling potential lattice-based attacks
Code-based attacks	Code-based cryptographic schemes	There are concerns about potential attacks on the underlying code structures, such as the Goppa code, which is used in some code-based cryptographic systems	Train a BayesNet model on features characterizing the coding structures in cryptographic systems. Evaluate the probability of observed patterns suggesting potential code-based attacks.
Multivariate polynomial attacks	Multivariate Polynomial-based cryptographic schemes	Attackers may attempt to find efficient algorithms for solving polynomial systems	Utilize a BayesNet model trained on features describing behaviour of multivariate polynomial-based cryptographic systems. Assess the probability of observed patterns indicative of potential attacks
Hash-based attacks	Hash-based cryptographic schemes (e.g., Merkle hash trees for digital signatures)	Potential vulnerabilities in specific constructions	Train a BayesNet model on features associated with hash function outputs and usage. Calculate the probability of observed pattersignalling potential hash-based attacks

Table 8
Detection results from different datasets.

Parameter	Generated	KDD-CUP99	CIC-MalMem-2022	ToN-IoT
Model build time (seconds)	0.3	0.77	0.08	3.63
Model test time (seconds)	0.27	98.83	16.25	0.63
Total Instances Tested	209338	148206	17579	119137
Correctly Classified	209334	148116	17526	117663
Correctly Classified %	99.9981	99.9393	99.6985	98.7628
Incorrectly classified	4	90	53	1474
Incorrectly classified %	0.0019	0.0607	0.3015	1.2372
Kappa Statistics	1	0.9981	0.994	0.9506
Mean Error (Absolute)	0	0.001	0.0029	0.0039
Root Mean Squared Error	0.0035	0.0245	0.0533	0.0502
Relative Absolute Error %	0.009	0.3207	0.5818	6.2902

Table 9
Prevention results from different datasets.

Metric	Generated	KDD-CUP99	CIC-MalMem-2022
Sensitivity	100.00%	98.53%	99.35%
Specificity	0.20%	99.05%	98.99%
Positive predictive value	99.56%	99.76%	99.00%
Negative predictive value	54.55%	94.29%	99.35%
False + rate for true $\tilde{D}$	99.80%	0.95%	1.01%
False + rate for true D	0.00%	1.47%	0.65%
False + rate for classified +	0.44%	0.24%	1.00%
False + rate for classified −	45.45%	5.71%	0.65%
Correctly Classified	99.56%	98.63%	99.17%

Table 10
Comparison of PETRAK with existing models.

Reference(s)	Detection type	ML	Dataset	Accuracy
Zang et al. (2021) [10]	Anomaly-based	Random Forest	Mininet-Wifi and CIC-IDS 2017	95%
Bangui et al. (2021) [11]	Hybrid (Anomaly, Signature)	Random Forest & K-means	CIC-IDS 2017	96%
Goncalves et al. (2021) [12]	Anomaly	Several algorithms	Public Vehicular dataset	96%
Balyan et al. (2022) [31]	Anomaly	Random Forest	NSL-KDD	98%
Soni et al. (2022) [13]	Behaviour/ reputation of attacker	Novel secure IPS algorithm	Driver behaviour is tested	89%
Malik et al. (2022) [14]	Anomaly/ node behaviour	Novel DPBHA	Driver behaviour is tested	95%
Rashid et al. (2023) [32]	Node behaviour	Distributed multi-layer classifier	Real-time generated dataset	94%-99%
Magsi et al. (2023) [33]	Anomaly based	Binary classification	NDN-IFA-Feature Selection	94%
Proposed PETRAK	Anomaly and Signature	Logit Model, BayesNet	Simulated, KDDCUP 99, CIC MalMem	99%

As the landscape of cybersecurity continues to evolve, future research avenues emerge to enhance the cryptographic resilience of DDoS detection systems. One potential future direction involves the integration of post-quantum cryptography (PQC) algorithms into DDoS detection mechanisms. Exploring how PQC can mitigate potential threats in the quantum computing era and adapting these algorithms to resource-constrained environments, such as the Cortex-M4 processor, presents an exciting challenge.

Another promising avenue is the investigation of machine learning and artificial intelligence techniques for anomaly detection in DDoS scenarios. Leveraging advanced algorithms to discern subtle patterns indicative of DDoS attacks can significantly bolster the accuracy and efficiency of detection systems. Future research could focus on the seamless integration of cryptographic methods with machine learning for a more comprehensive and adaptive defence against evolving DDoS threats.

Moreover, exploring the intersection of lightweight cryptography and DDoS resilience is paramount. Investigating how lightweight cryptographic algorithms, designed for efficiency in constrained environments, can be tailored to fortify DDoS detection mechanisms would be a valuable research trajectory.

CRediT authorship contribution statement

Amandeep Verma: Writing – review & editing, Writing – original draft, Software, Data curation, Conceptualization. **Rahul Saha:** Writing – review & editing, Writing – original draft, Validation, Methodology, Conceptualization. **Gulshan Kumar:** Writing – review & editing, Validation, Methodology. **Mauro Conti:** Writing – review & editing, Validation, Supervision, Methodology.

Declaration of competing interest

On behalf of all the authors, I declare that we do not have any known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Arbor Networks, Cisco annual internet report, 2018–2023, 2022, available at https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf. (Accessed on 8 September 8 September 2022).
[2] R. Kitchin, M. Dodge, The (in) security of smart cities: Vulnerabilities, risks, mitigation, and prevention, J. Urban Technol. 26 (2) (2019) 47–65.
[3] A. Ioanid, C. Scarlat, G. Militaru, The effect of cybercrime on Romanian SMEs in the context of wannacry ransomware attacks, in: European Conference on Innovation and Entrepreneurship, 2017, pp. 307–313.
[4] S. Soderi, D. Masti, Y.Z. Lun, Railway cyber-security in the era of interconnected systems: A survey, IEEE Trans. Intell. Transp. Syst. 24 (7) (2023) 6764–6779.
[5] J. Anna, 2022, available on https://cybernews.com/news/a-suspected-cyberattack-on-italys-state-railway-disrupts-ticket-sales/. (Accessed on 10 September 2022),
[6] Z. Wang, X. Liu, Cyber security of railway cyber–physical system (CPS)–A risk management methodology, Commun. Transp. Res. 2 (2022) 1–11.
[7] A. Verma, R. Saha, N. Kumar, G. Kumar, A detailed survey of denial of service for IoT and multimedia systems: Past, present and futuristic development, Multimedia Tools Appl. 81 (14) (2022) 19879–19944.
[8] M. De Donno, N. Dragoni, A. Giaretta, A. Spognardi, DDoS-capable IoT malware: Comparative analysis and mirai investigation, Secur. Commun. Netw. 2018 (2018) 1–30.
[9] A. Verma, R. Saha, G. Kumar, T.H. Kim, The security perspectives of vehicular networks: A taxonomical analysis of attacks and solutions, Appl. Sci. 11 (10) (2021) 1–25.
[10] M. Zang, Y. Yan, Machine learning-based intrusion detection system for big data analytics in VANET, in: IEEE 93rd Vehicular Technology Conference, (VTC2021-Spring), Helsinki, Finland, 2021, pp. 1–5.
[11] H. Bangui, M. Ge, B. Buhnova, A hybrid data-driven model for intrusion detection in VANET, Procedia Comput. Sci. 184 (2021) 516–523.
[12] F. Gonçalves, J. Macedo, A. Santos, An intelligent hierarchical security framework for vanets, Information 12 (11) (2021) 1–26.
[13] G. Soni, K. Chandravanshi, M.K. Jhariya, A. Rajput, An IPS approach to secure V-RSU communication from blackhole and wormhole attacks in VANET, in: Contemporary Issues in Communication, Cloud and Big Data Analytics: Proceedings of CCB 2020, 2022, pp. 57–65.
[14] A. Malik, M.Z. Khan, M. Faisal, F. Khan, J.T. Seo, An efficient dynamic solution for the detection and prevention of black hole attack in vanets, Sensors 22 (5) (2022) 1–27.
[15] O.K. Nicesio, A.G. Leal, V.L. Gava, Quantum machine learning for network intrusion detection systems, A systematic literature review, in: IEEE 2nd International Conference on AI in Cybersecurity, (ICAIC), Houston, TX, USA, 2023, pp. 1–6.
[16] M. Kalinin, V. Krundyshev, Security intrusion detection using quantum machine learning techniques, J. Comput. Virol. Hacking Tech. 19 (1) (2023) 125–136.
[17] D.B. Salvakkam, V. Saravanan, P.K. Jain, R. Pamula, Enhanced quantum-secure ensemble intrusion detection techniques for cloud based on deep learning, Cogn. Comput. 15 (5) (2023) 1–20.
[18] D. Lightbody, D.M. Ngo, A. Temko, C.C. Murphy, E. Popovici, Attacks on IoT: Side-channel power acquisition framework for intrusion detection, Future Internet 15 (5) (2023) 1–27.
[19] R. Oshana, M.A. Thornton, M. Caraman, A side channel attack detection system using processor core events and a support vector machine, in: 11th Mediterranean Conference on Embedded Computing, (MECO), Budva, Montenegro, 2022, pp. 1–8.
[20] A. Mila, R. Azarderakhsh, M.M. Kermani, L. Beshaj, Time-efficient finite field microarchitecture design for curve448 and Ed448 on Cortex-M4, in: International Conference on Information Security and Cryptology, Springer Nature, Cham, Switzerland, 2022, pp. 292–314.
[21] A. Mila, R. Azarderakhsh, M.M. Kermani, Fast strategies for the implementation of SIKE round 3 on ARM cortex-M4, IEEE Trans. Circuits Syst. I. Regul. Pap. 68 (10) (2021) 4129–4141.
[22] S. Pakize, E. Karagoz, H. Seo, R. Azarderakhsh, M.M. Kermani, International Conference on Security and Privacy in Communication Systems, Springer International Publishing, Cham, 2021, pp. 424–440,
[23] B. Niasar, M.R. Azarderakhsh, M.M. Kermani, Cryptographic accelerators for digital signature based on Ed25519, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 29 (7) (2021) 1297–1305.
[24] J. Amir, R. Azarderakhsh, M.M. Kermani, D. Jao, Supersingular isogeny Diffie–Hellman key exchange on 64-bit ARM, IEEE Trans. Dependable Secure Comput. 16 (5) (2017) 902–912.
[25] J. Kaur, A. Sarker, M.M. Kermani, R. Azarderakhsh, Hardware constructions for error detection in lightweight Welch-Gong (WG)-oriented streamcipher WAGE benchmarked on FPGA, IEEE Trans. Emerg. Top. Comput. 10 (2) (2021) 1208–1215.
[26] M.M. Kermani, R. Azarderakhsh, J. Xie, Error detection reliable architectures of Camellia block cipher applicable to different variants of its substitution boxes, in: 2016 IEEE Asian Hardware-Oriented Security and Trust, (AsianHOST), IEEE, 2016, pp. 1–6.
[27] A. Anita, M.M. Kermani, R. Azarderakhsh, Fault diagnosis schemes for low-energy block cipher midori benchmarked on FPGA, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (4) (2016) 1528–1536.
[28] J. Smith, A. Johnson, Block cipher QARMA with error detection mechanisms, in: Proceedings of the IEEE International Conference on Cryptography, London, UK, 2023, pp. 29–30.
[29] S.J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P.K. Chan, Cost-based modelling for fraud and intrusion detection: results from the JAM project, in: Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00, vol. 2, 2000, pp. 130–144.
[30] T. Carrier, P. Victor, A. Tekeoglu, A.H. Lashkari, Detecting obfuscated malware using memory feature engineering, in: 8th International Conference on Information Systems Security and Privacy, (ICISSP), 2022, pp. 1–12.
[31] A.K. Balyan, S. Ahuja, U.K. Lilhore, S.K. Sharma, P. Manoharan, A.D. Algarni, K. Raahemifar, A hybrid intrusion detection model using ega-pso and improved random forest method, Sensors 22 (16) (2022) 1–20.
[32] R. Kanwal, Y. Saeed, A. Ali, F. Jamil, R. Alkanhel, A. Muthanna, An adaptive real-time malicious node detection framework using machine learning in vehicular ad-hoc networks (VANETs), Sensors 23 (5) (2023) 1–34.
[33] A.H. Magsi, S.A.H. Mohsan, G. Muhammad, S. Abbasi, A machine learning-based interest flooding attack detection system in vehicular named data networking, Electronics 12 (18) (2023) 1–19.