UDC 004.056.5
https://doi.org/10.32362/2500-316X-2025-13-5-25-40
EDN JKQMQM

RESEARCH ARTICLE

Simulation model of a scalable method for detecting multi-vector attacks taking into account the limitations of computing and information resources of IoT devices

Vyacheslav I. Petrenko, Fariza B. Tebueva, Maxim G. Ogur @, Gennady I. Linets, Valery P. Mochalov

North Caucasus Federal University, Stavropol, 355017 Russia
@ Corresponding author, e-mail: ogur26@gmail.com

• **Submitted:** 14.10.2024 • **Revised:** 13.05.2025 • **Accepted:** 07.08.2025

Abstract
Objectives. The study sets out to develop a scalable method for detecting multi-vector attacks on Internet of Things (IoT) devices. Given the growth of security threats in IoT networks, such a solution must provide high accuracy in detecting attacks with minimal computing costs while taking into account the resource constraints of IoT devices.
Methods. The developed hybrid neural network architecture combines convolutional networks for spatial dependence analysis and long short-term memory networks or gated recurrent units representing types of recurrent neural networks for analyzing time dependencies in network traffic. Model parameters and computational costs are reduced by pruning. A blockchain with a proof of voting[1] consensus mechanism provides secure data management and decentralized verification.
Results. Experiments on the CIC IoT Dataset 2023[2] showed the effectiveness of the model: the accuracy and F1 measure were 99.1%. This confirms the ability to detect known and new attacks in real time with high accuracy and completeness. Processing time is reduced to 12 ms, while memory usage is reduced to 180 MB, which makes the model suitable for devices with limited resources.
Conclusions. The developed model is superior to analogues in terms of accuracy, processing time, and memory usage. Hybrid architecture, pruning, and decentralized verification provide effectiveness against multi-vector IoT threats.

Keywords: multi-vector attacks, Internet of Things, threat detection, neural networks, blockchain, neuronal pruning, cybersecurity, node compromise, consensus, federated learning

[1] Proof of Voting is a consensus algorithm in blockchain networks, in which participants confirm transactions and ensure network security by voting for blocks or transactions.
[2] CIC IoT Dataset 2023. http://cicresearch.ca/IOTDataset/CIC_IOT_Dataset2023/Dataset/. Accessed June 30, 2025.

For citation: Petrenko V.I., Tebueva F.B., Ogur M.G., Linets G.I., Mochalov V.P. Simulation model of a scalable method for detecting multi-vector attacks taking into account the limitations of computing and information resources of IoT devices. *Russian Technological Journal.* 2025;13(5):25–40. https://doi.org/10.32362/2500-316X-2025-13-5-25-40, https://www.elibrary.ru/JKQMQM

Financial disclosure: The authors have no financial or proprietary interest in any material or method mentioned.

The authors declare no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

Имитационная модель масштабируемого метода выявления многовекторных атак с учетом ограничений вычислительных и информационных ресурсов IoT-устройств

В.И. Петренко, Ф.Б. Тебуева, М.Г. Огур @, Г.И. Линец, В.П. Мочалов

Северо-Кавказский федеральный университет, Ставрополь, 355017 Россия
@ Автор для переписки, e-mail: ogur26@gmail.com

• **Поступила:** 14.10.2024 • **Доработана:** 13.05.2025 • **Принята к опубликованию:** 07.08.2025

Резюме
Цели. Основная цель работы – разработка масштабируемого метода для выявления многовекторных атак на устройства интернета вещей (Internet of Things, IoT). Учитывая рост угроз безопасности в IoT-сетях, решение должно обеспечивать высокую точность обнаружения атак при минимальных вычислительных затратах и с учетом ограничений ресурсов IoT-устройств.
Методы. Для достижения поставленной цели разработана гибридная архитектура нейронных сетей, сочетающая сверточные сети для анализа пространственных зависимостей и сети долгой краткосрочной памяти или Gated Recurrent Units (управляемые рекуррентные блоки) – один из видов рекуррентных нейронных сетей для анализа временных зависимостей в сетевом трафике. Техника обрезки (pruning) сокращает параметры модели и вычислительные затраты. Блокчейн с механизмом консенсуса Proof of Voting[3] обеспечивает безопасное управление данными и децентрализованную верификацию.
Результаты. Эксперименты на датасете CIC IoT Dataset 2023[4] показали эффективность модели: точность и F1-мера составили 99.1%, что подтверждает способность выявлять известные и новые атаки в реальном времени с высокой точностью и полнотой. Время обработки сокращено до 12 мс, использование памяти – до 180 МБ, что делает модель пригодной для устройств с ограниченными ресурсами.
Выводы. Разработанная модель превосходит аналоги по точности, времени обработки и использованию памяти. Гибридная архитектура, обрезка и децентрализованная верификация обеспечивают эффективность против многовекторных угроз IoT. Работа открывает перспективы для исследований в кибербезопасности, предлагая решения для защиты IoT-сетей от сложных атак.

[3] Proof of Voting (алгоритм консенсуса) – это консенсусный алгоритм в блокчейн-сетях, при котором участники подтверждают транзакции и обеспечивают безопасность сети путем голосования за блоки или транзакции. [Proof of Voting is a consensus algorithm in blockchain networks, in which participants confirm transactions and ensure network security by voting for blocks or transactions.]

[4] CIC IoT Dataset 2023. http://cicresearch.ca/IOTDataset/CIC_IOT_Dataset2023/Dataset/. Дата обращения 30.06.2025. / Accessed June 30, 2025.

Ключевые слова: многовекторные атаки, интернет вещей, выявление угроз, нейронные сети, блокчейн, обрезка нейронов, кибербезопасность, компрометация узлов, консенсус, федеративное обучение

Для цитирования: Петренко В.И., Тебуева Ф.Б., Огур М.Г., Линец Г.И., Мочалов В.П. Имитационная модель масштабируемого метода выявления многовекторных атак с учетом ограничений вычислительных и информационных ресурсов IoT-устройств. *Russian Technological Journal*. 2025;13(5):25–40. https://doi.org/10.32362/2500-316X-2025-13-5-25-40, https://www.elibrary.ru/JKQMQM

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

INTRODUCTION

Due to the development of Internet of Things (IoT) technologies networks of IoT devices have become an integral part of modern informational infrastructure. These devices ensure the interaction of numerous systems and platforms in real time to increase the effectiveness, convenience, and flexibility of various sectors: from smart homes and cities to industrial and medical systems. However, IoT devices are often limited in computational and energy resources, making them sensitive for multi-vector cyberattacks. Thus, their wide distribution is associated with an increase in the number of potential threats to information security. Among the most dangerous attacks are DDoS[5], routing attacks, SQL[6] injections, and other forms of multi-vector threats.

Modern methods of detecting attacks, which typically require significant computational resources, can prove inefficient under limited conditions of IoT. This leads to the necessity to develop new approaches that take into account the limitations of computational and informational resources of IoT and simultaneously ensure high security level.

In the present work, we propose a scalable model for detecting multi-vector attacks, which represents a hybrid architecture of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRM) neural networks (CNN + LSTM/GRU[7]) for analyzing spatiotemporal dependencies of network traffic and decentralized data verification. In order to reduce computational costs, the model incorporates the use of blockchain technologies and neuron pruning. The proposed model, which is oriented towards working in real time given limited resources, is applicable for modern IoT networks. The experimental testing of the efficiency of the developed model using the CIC IoT Dataset 2023 dataset demonstrated its superiority over existing solutions.

[5] Distributed Denial of Service is a form of cyberattack on web systems in order to disable them or make it difficult for ordinary users to access them.

[6] Structured Query Language.

[7] Gated Recurrent Unit.

1. ANALYSIS OF LITERATURE

Sen et al. [1] proposed a mathematical tool for modeling cyberattacks on electric grids involving game theory and the construction of attack graphs. The model is underlaid by the concept of attacker–defender dynamics, where the attacker tries to damage the operation of an electric grid, while the defender tries to prevent the damage using proactive and reactive defense measures. The main advantage of the model is taking into account the attacker–defender dynamics, which makes the model more realistic for use in complex systems. The model uses attack graphs to model multilayer and multistep attacks, taking into account their complexity and diversity. However, the drawbacks of the model include the requirement of initial data on the system and its vulnerabilities, as well as the requirement of the exact evaluation of probabilities of a successful attack and cyber-shutdown cost. This may complicate its practical application for limited data systems.

Lysenko et al. [2] proposed a method for detecting multi-vector cyberattacks on IoT infrastructure by analyzing network traffic and machine learning. Lysenko et al. distinguished four key types of signs helping in accelerated detection of attacks based on data flows, MQTT[8], DNS[9], and HTTP[10]. The method helps increasing the efficiency of detecting attacks by early diagnosis of harmful traffic by analyzing flows and deep analyzing packets for exact detection of multi-vector attacks. This makes this method available for IoT networks with high data volume and complex attack structure. However, the complexity of the method is due to the necessity for an exact determination of a set of signs and their processing, which requires much real-time computational resources in large IoT networks.

Aguru and Erukala [3] proposed a methodology of protecting decentralized IoT networks from multi-vector DDoS attacks using blockchain technologies and deep learning. A Prevent-then-Detect two-stage approach was proposed, where, at the first stage, an intrusion

[8] Message queuing telemetry transport.

[9] Domain name system.

[10] HyperText transfer protocol.

prevention system (IPS) works through a blockchain consortium of validators, while at the second stage, an intrusion detection system (IDS) uses deep learning models for analyzing network traffic and detecting threats. The blockchain ensures security of data transfer between network nodes and controls access to resources of IoT network using intellectual contracts, which are used to determine actions on attack detection and threat prevention. The attack prevention system in a blockchain consortium uses a consensus algorithm to test suspicious traffic. However, the significant computational resources required for the operation of blockchain system and deep neural networks may limit its operability in devices with limited computational possibilities in IoT networks.

Ipole-Adelaiye et al. [4] proposed a method for detecting multi-vector attacks (MVA) using a multilayer perceptron (MLP) approach to analyze network traffic by means of machine learning for detecting various attack vectors. In particular, network data from packet capturing (PCAP) are analyzed to determine anomalous patterns in the behavior of network compounds. The method uses neural networks for data classification and subsequent analysis to increase attack detection accuracy. The MLP, which represents the main component of the proposed system, consists of an input layer, a hidden layer, and an output layer. While the MLP method is suitable for the problems where a high detection accuracy is important, faster models can be used for networks with limited computational resources.

Pakmehr et al. [5] analyze various methods for detecting DDoS attacks on IoT networks with an emphasis on features and challenges that emerge when applying these methods to IoT networks. The authors reviewed several categories of attack detection methods, including signature, anomalous, and hybrid approaches. The mathematical tool includes algorithms based on machine learning, such as Support Vector Machine (SVM), Decision Trees, K-Nearest Neighbors (KNN), and Random Forest. These algorithms are used to classify network traffic and separate anomalous patterns characteristic of DDoS attacks. Although the proposed approaches more efficiently cope with high-volume and various attacks on IoT devices, their efficiency is limited by the complexity of refining the models and the necessity of large computational resources.

Alhakami [6] proposed a mathematical tool for estimating invasion detection methods under Gen V Multi-Vector Attacks. The method is based on a combination of two methods: Fuzzy Analytic Hierarchy Process (Fuzzy AHP) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS). These methods allow the estimation of various criteria of efficiency of attack detection systems: detection accuracy, adaptivity, scaling, effect on resources, detection time, and automation. A special attention is made to such aspects as adaptation to new threats, the possibility of operating in scaling networks, and the minimization of loading resources at high automation and rapid response.

Saiyed and Al-Anbagi [7] propose an approach for detecting multi-vector DDoS attacks on IoT networks using deep ensemble learning with pruning. Saiyed and Al-Anbagi presented the Deep Ensemble learning with Pruning (DEEPShield) system, which combines the CNN and LSTM networks for analyzing network traffic and detecting both high-volume, and low-volume DDoS attacks. The mathematical tool is based on using an ensemble approach, where CNN extracts spatial signs from network traffic, and LSTM is used to analyze time dependencies. The DEEPShield system demonstrates a high (>90%) accuracy of attack detection and reduces prediction time in comparison with similar models.

Doe et al. [8] describe a hybrid model for analyzing threats and classifying attacks in IoT networks using deep learning and adaptive optimization algorithm Mayfly (LAMOA[11]). The model tends to the detection of routing attacks on IoT networks (sinkhole, wormhole, black hole, and Sybil), which significantly reduce their productivity and security. The model is based on recurrent neural network with long short-term memory for processing time series of network traffic and classification of attacks with the Mayfly adaptive algorithm for optimizing of model hyperparameters. The model, which demonstrates a high ability to exact classification of various types of attacks, is an efficient solution for ensuring security of IoT networks; however, its complexity and computational costs may limit its use in networks with limited sources, thus requiring its further optimization.

Aguru and Erukala [9] propose a lightweight framework for detecting multi-vector DDoS attacks on mobile medicine networks based on IoT using deep learning. Here, the focus on accuracy and efficiency aligns with the purposes of the proposed model and underscores the need for adaptation of detection methods to the specificity of mobile IoT devices, which makes their operation actual for further research in this area.

Petrenko et al. [10] present a method for detecting and counteracting multi-vector threats on decentralized IoT systems that emphasizes the necessity of complex security strategies. The authors underline the importance of integrating various protection methods, including machine learning and blockchain technologies.

The works [11–15] compare approaches to common mitigation of attacks on cloud and fuzzy

[11] Learning-based Adaptive Mayfly Optimization Algorithm.

computations, which may improve the scalability of the model developed in the work, Leng et al. [11] propose methods for improving the scalability and efficiency of protection of IoT networks, which is an important aspect of ensuring security in conditions of growing number of devices and traffic volume. Ali et al. [12] describe a method for protecting decentralized IoT networks from multi-vector DDoS attacks using blockchain technologies and deep learning methods. The proposed two-stage approach combines prevention and detection of attacks, allowing for the efficient control of network threats and an improved level of security. Dalal et al. [13] propose a method for detecting multi-vector attacks based on MLP. Attention is focused on the importance of analysis of network traffic for identifying anomalous patterns, which is a key moment for improving attack detection accuracy. Zahid et al. [14] review various methods for detecting DDoS attacks on IoT networks. The authors discuss signature-, anomalous-, and hybrid approaches in terms of their advantages and disadvantages in the context of IoT. Lungu et al. [15] describe a mathematical tool for estimating invasion detection methods in conditions of fifth-generation multi-vector attacks. By combining decision-making methods, the efficiency of attack detection systems can be estimated, including accuracy and adaptivity.

To construct a simulation model of a scalable method of detection of multi-vector attacks taking into account the limitations of computational and information resources of IoT devices, the following three most suitable models were chosen:

(1) Deep Ensemble Learning with Pruning method [7] using a combination of CNN and LSTM for analyzing network traffic and pruning to reduce computational costs.
(2) Threat Analysis model [8] using LSTM for analyzing routing attacks on IoT networks with adaptive optimization of hyperparameters by the Mayfly algorithm.
(3) Blockchain-based Threat Intelligence Framework method [3] combining blockchain technology with deep learning for protection of IoT networks from multi-vector DDoS attacks.

2. SIMULATION MODEL OF SCALABLE METHOD OF DETECTING MULTI-VECTOR ATTACKS TAKING INTO ACCOUNT LIMITATIONS OF COMPUTATIONAL AND INFORMATIONAL RESOURCES OF IoT DEVICES

The developed model, which should detect multi-vector attacks with high accuracy while minimizing computational costs and taking into account limitations inherent in IoT devices, must be suitable for scaling in large decentralized IoT networks.

The simulation model is built from the following main components:

(1) CNN + LSTM/GRU network traffic analysis module.
(2) Mayfly algorithm for adaptive optimization of hyperparameters.
(3) Proof of Voting (PoV) blockchain-oriented consensus mechanism for decentralized verification.
(4) Neuron pruning for reducing computational costs.

Let us consider these components in more detail.

2.1. CNN + LSTM/GRU network traffic analysis module

The module for network traffic analysis and detection of anomalies in data sequence uses a hybrid architecture of CNN and LSTM (or GRU to reduce computing costs), where:

- The CNN processes spatial signs of network traffic. Input data are represented as a multidimensional tensor, where each element characterizes network packets (e.g., time, size, protocol type). Convolutional layers separate spatial patterns in traffic;
- The LSTM (or GRU) analyzes time dependencies. This helps detect complex multi-vector attacks, which manifests themselves on different time range. Long Short-Term Memory shares information on the previous states of traffic and helps predicting future events, which is important for detecting long-term attacks, such as DDoS.

The network traffic analysis module in the simulation model is based on a hybrid architecture, which combines convolutional neural networks for analysis of spatial dependencies of network traffic and recurrent neuronal networks (LSTM or GRU) for analysis of time dependencies. Such a structure efficiently analyzes multi-vector attacks, which may reveal themselves through complex anomalies in spatial and time dependencies of network traffic.

The convolutional neural network is used to separate spatial signs from network traffic represented by multidimensional data (tensor). The input traffic, including such parameters as time steps, packet size, protocol type, IP addresses, and other metrics, is transformed into a tensor of dimension $\mathbf{X} \in \mathbb{R}^{h \times w \times c}$, where h is the tensor height (number of packets or time steps), w is the tensor width (number of signs or characteristics per packet), and c is the number of channels (e.g., this may be separation by protocols of data types).

The man convolutional equation is written as

$$\mathbf{Y}_{i,j,k} = \sum_{m=1}^{h_k} \sum_{n=1}^{w_k} \mathbf{X}_{i+m,j+n,c} \mathbf{W}_{m,n,k} + b_k, \qquad (1)$$

where $\mathbf{X}_{i,j,c}$ is the input tensor for position (i, j) in the channel c, $\mathbf{W}_{m,n,k}$ is the convolutional filter of sizes

$h_k \times w_k$ for the channel k, b_k is the bias for the channel k, and $\mathbf{Y}_{i,j,k}$ is the convolutional result in the channel k.

Convolution (1) is followed by applying an activation function for increasing nonlinearity:

$$\mathbf{Z}_{i,j,k} = \mathrm{ReLU}(\mathbf{Y}_{i,j,k}) = \max(0, \mathbf{Y}_{i,j,k}).$$

Here, ReLU (Rectified Linear Unit) is one of the most popular activation functions, which retains only positive values.

Convolution network separates spatial patterns in network traffic data, such as packet frequency and correlation of various traffic parameters. After spatial signs are isolated using CNN, they are transferred to LSTM to analyze time dependencies. Long Short-Term Memory takes into account the time behavior of traffic and helps isolating multi-vector attacks, which may show through sequential changes in network behavior.

Let us consider the main components of LSTM,

(1) Input gates controls the choice of a new input state to refresh state of memory. The input gates at time t are activated as

$$i_t = \sigma(\mathbf{W}_{\mathrm{in}} \cdot [h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{in}}), \qquad (2)$$

where $\mathbf{x}_t$ is the input vector at time t (spatial signs isolated from CNN); h_{t-1} is the hidden state at the previous time step, $\mathbf{W}_{\mathrm{in}}$ is the weight matrix for the input gate, $\mathbf{b}_{\mathrm{in}}$ is the bias vector for the input gate, and σ is a sigmoid normalizing values on the interval [0, 1].

(2) Forgetting gates determine the part of the previous state that should be preserved. The function f_t for activation of forgetting gates can be written as

$$f_t = \sigma(\mathbf{W}_{\mathrm{f}} \cdot [h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{f}}), \qquad (3)$$

where $\mathbf{W}_{\mathrm{f}}$ is the weight matrix for the forgetting gate; $\mathbf{b}_{\mathrm{f}}$ is the bias vector for the forgetting gate.

(3) State of memory C_t is refreshed at each time step taking into account new information as:

$$C_t = f_t\, C_{t-1} + i_t \tanh(\mathbf{W}_{\mathrm{c}} \cdot [h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{c}}), \qquad (4)$$

where C_{t-1} is the previous state of memory; C_t is the new state of memory; f_t is the forgetting gates; i_t is the input gates; tanh is the hyperbolic tangent, which is a function of activation used for refreshing state of memory; $\mathbf{W}_{\mathrm{c}}$ is the weight matrix for state of memory; $[h_{t-1}, \mathbf{x}_t]$ is the concatenation of the hidden state at the previous step and the current input vector; and $\mathbf{b}_{\mathrm{c}}$ is the bias vector for refreshing state of memory; the index c indicates the collection of data to memory.

(4) Output gates control the part of state of memory that should be used for refreshing the hidden state. The output gates are activated as follows:

$$o_t = \sigma(\mathbf{W}_{\mathrm{o}} \cdot [h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{o}}), \qquad (5)$$

where o_t is the activation function of the output gates, σ is the activation sigmoid, $\mathbf{W}_{\mathrm{o}}$ is the weight matrix for the output gates, and $\mathbf{b}_{\mathrm{o}}$ is the bias vector for the output gates. The new hidden state h_t is calculated as

$$h_t = o_t \tanh(C_t), \qquad (6)$$

where h_t is the hidden state at time t, which is used for final classification of network traffic; and C_t is the current state of memory.

Instead of LSTM, GRU can be used, which is modification that uses less computational costs. GRU combines forgotten and input gates into a single refreshing gate, which reduces computational costs and improves model operation under limited resources.

Let us consider the main components of GRU.

(1) Refreshing gates:

$$z_t = \sigma(\mathbf{W}_{\mathrm{z}} \cdot [h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{z}}). \qquad (7)$$

Here, z_t is the activation of the refreshing gates, which controls how strongly the current state affects the previous one; the index z shows a refreshing gate (zero).

(2) Removal gates:

$$r_t = \sigma(\mathbf{W}_{\mathrm{r}} \cdot [h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{r}}). \qquad (8)$$

Here, r_t is the activation of the removal gates, which controls how strongly the previous state should be forgotten; the index r indicates the reset gate.

(3) Refreshing of the hidden state:

$$h_t = (1 - z_t)\, h_{t-1} + z_t \tanh(\mathbf{W}_{\mathrm{h}}[\mathbf{r}_t h_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathrm{h}}), \qquad (9)$$

where h_t is the refreshed hidden state at time t, and $\mathbf{W}_{\mathrm{h}}$ is the weight matrix for refreshing the hidden state; the index h displays the gate of hidden state.

Recurrent neural network GRU uses fewer parameters than LSTM, which makes it more suitable for problems requiring smaller computational costs, such as operation under limited resources of IoT devices.

After processing of the CNN and LSTM/GRU data, the model uses a fully connected layer for final classification of traffic. This layer calculates the probabilities that data belongs to one of the classes, e.g., normal traffic or attack:

$$P_{\mathrm{attack}} = \mathrm{Softmax}(\mathbf{W}_{\mathrm{out}} h_{\mathrm{T}} + \mathbf{b}_{\mathrm{out}}), \qquad (10)$$

where P_{attack} is the probability that the input traffic is attack, $\mathbf{W}_{\mathrm{out}}$ is the outer layer weights, h_{T} is the hidden state at the last time step, $\mathbf{b}_{\mathrm{out}}$ is the outer layer bias, and Softmax normalizes the outer probability values.

The main variables of the model are the input tensor $\mathbf{X} \in \mathbb{R}^{h \times w \times c}$ of network traffic; the weight matrices **W** of layers (CNN, LSTM/GRU); the bias vectors **b** of layers (CNN, LSTM/GRU); the input (i_t), forgotten (f_t), and output (o_t) gates to LSTM, respectively; the refreshing (z_t) and removal (r_t) gates to GRU, respectively; the hidden state h_t at time t; the state of memory C_t in LSTM; and the probability P_{attack} that the traffic is attacking.

The network traffic analysis module based on a hybrid architecture of CNN + LSTM (or GRU) combines spatial and time dependencies of network data. Such architecture efficiently detects multi-vector attacks on IoT networks, which is particularly important for systems with limited computational resources.

The model operates as follows.

(1) The input traffic is transformed into a multidimensional tensor, which enters convolutional layers for isolating signs.

(2) The isolated signs enter the LSTM/GRU for analyzing time dependencies.

(3) The model classifies the traffic as normal or attacking.

2.2. Adaptive optimization of hyperparameters using the Mayfly adaptive algorithm

To improve the efficiency of the model and tune it to specific network conditions (e.g., data volume or type of attacks), we use the Mayfly adaptive algorithm [8]. The Mayfly algorithm helps to automatically find the optimal hyperparameters of the model, such as:

- number of layers in CNN and LSTM;
- number of filters and neurons in each layer;
- model learning rate.

The Mayfly adaptive algorithm accelerates model tuning and ensures its optimal productivity without necessity of manual tuning. The algorithm uses evolutionary methods for searching for optimal parameters and is adapted during model learning.

The main steps of the Mayfly adaptive algorithm are the following:

1. Population initialization.
2. Males and females: separation into two groups with different search strategies.
3. Global and local search: search for the best solutions by males and females.
4. Evolution and refreshment of rates and positions.
5. Rendezvous and reproduction.

The main variables and parameters of the algorithm are the following.

N—number of individuals in the population;

x_i^{m}—position of male i in solution space (hyperparameter value);

x_i^{f}—position of female I;

v_i^{m}—male velocity;

v_i^{f}—female velocity;

α, β, γ—male and female motion control coefficients (inertia, acceleration, and interaction, respectively);

g_{best}—global best solution found by all males and females;

p_{best}—personal best solution found by male or female;

λ—attraction coefficient for rendezvous of males and females;

∊—random bias affecting mutation in search.

2.2.1. Population initialization

The Mayfly algorithm begins with random initialization of the initial population of males and females (hyperparameters) in search space. Each male or female is a model hyperparameter vector

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}],$$

where d is the hyperparameter space dimensionality (e.g., number of layers or neurons, training rate, etc.).

The position x_i of each male or female in hyperparameter space is initialized randomly:

$$\mathbf{x}_i^{\text{m}}(0), \mathbf{x}_i^{\text{f}}(0) \sim \text{Uniform}(\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{max}}),$$

where $\mathbf{x}_{\text{min}}$ and $\mathbf{x}_{\text{max}}$ are the hyperparameter space boundaries, and Uniform($\mathbf{x}_{\text{min}}$, $\mathbf{x}_{\text{max}}$) is the function of random sampling from the interval [$\mathbf{x}_{\text{min}}$, $\mathbf{x}_{\text{max}}$].

2.2.2. Refreshment of male velocity and position

Males find solutions in global space, refreshing their position based on personal best solution p_{best} and global best solution g_{best}. The male position refreshment velocity is calculated as:

$$\mathbf{v}_i^{\text{m}}(t+1) = \alpha \mathbf{v}_i^{\text{m}}(t) + \beta_1 r_1 (p_{\text{best},i} - \mathbf{x}_i^{\text{m}}(t)) + \\ + \beta_2 r_2 (g_{\text{best},i} - \mathbf{x}_i^{\text{m}}(t)),$$

where α is the inertia coefficient (which controls how strongly the velocity of the previous step affects the current position); β_1 and β_2 are the acceleration coefficients, which control the effect of the personal and global bets solutions on the velocity refreshment; and r_1 and r_2 ~ Uniform(0, 1) are random values, which ensure random bias in search.

The position of each male is refreshed taking into account its new velocity:

$$\mathbf{x}_i^{\text{m}}(t+1) = \mathbf{x}_i^{\text{m}}(t) + \mathbf{v}_i^{\text{m}}(t+1).$$

2.2.3. Refreshment of female velocity and position

Females perform local search, refreshing their positions based on the distance to males. The female position refreshment velocity is calculated taking into account the interaction with males:

$$\mathbf{v}_i^{\mathrm{f}}(t+1) = \lambda(x_i^{\mathrm{m}}(t) - x_i^{\mathrm{f}}(t)) + \in,$$

where λ is the attraction coefficient between males and females, and ∈ is a random deviation for ensuring diversity of solutions.

The positions of females is refreshed as follows:

$$\mathbf{x}_i^{\mathrm{f}}(t+1) = \mathbf{x}_i^{\mathrm{f}}(t) + \mathbf{v}_i^{\mathrm{f}}(t+1).$$

2.2.4. Evaluation of solutions

Each male or female is evaluated using a fitness function, which may be related to model, accuracy, learning time, model complexity, and other parameters. The fitness function $F(\mathbf{x}_i)$ for each male or female is calculated as:

$$F(\mathbf{x}_i) = \text{Evaluation Model}(\mathbf{x}_i),$$

where $\mathbf{x}_i$ are the hyperparameters represented by male or female, and Evaluation Model is a function estimating the model productivity at the given hyperparameters.

2.2.5. Rendezvous and reproduction

After the velocities and positions are refreshed, males and females make a rendezvous, which models reproduction in the algorithm. When males and females become close enough, there are crossing over and mutation:

- crossing over transforms part of genetic information (hyperparameters) from males to females:

  $$\mathbf{x}_{\text{new}} = \lambda \mathbf{x}_i^{\mathrm{m}} + (1-\lambda)\mathbf{x}_i^{\mathrm{f}},$$

  where $\mathbf{x}_{\text{new}}$ is a new value (position) of male or female as acquired by crossing over, $\mathbf{x}_i^{\mathrm{m}}$ is the current position of male i, $\mathbf{x}_i^{\mathrm{f}}$ is the current position of female i, and λ is a coefficient determining the weight of influence of female in the new state (λ typically ranges from 0 to 1).
- mutation randomly changes some parameters with probability p_{mut}.

2.2.6. Completion criterion

The Mayfly algorithm performs until one of the following conditions is used: the maximum number $T_{\max}$ of iterations has reached, or the fitting function is not improved for several consecutive operations.

2.3. Decentralized verification mechanism based on blockchain technology

The security and reliability of the system under decentralized IoT networks are ensured by PoV blockchain-oriented consensus mechanism [3]. The main functions of this component are the following:

- decentralized verification of attack data, by which several network nodes analyze network traffic and send information on possible attacks to distributed ledger;
- validation of blocks occurs through polling of validating nodes: if more than 50% nodes support attack, the information about it is written in the distributed ledger, and dangerous IP addresses are blocked through action modules.

The decentralized verification module uses blockchain technologies for ensuring data security and preventing attacks in IoT networks. This module operates based on PoV consensus mechanism, which allows network nodes (validators) to vote for data blocks on traffic, attacks, or state of network. Blockchain ensures protection from deception of data, decentralized storage, and automatic performance of such actions as block of harmful IP addresses through action modules.

The main elements and variables are the following.

B—data block containing information on network traffic, detected attacks, or refreshing states of network;

N—number of nodes (validators) in blockchain network;

V_i—voice of validator *i* for acceptance or rejection of block;

P_{valid}—probability that the block is valid;

T_B—block validation time;

AM—Action Module, which contains data on attacks and blocks of IP addresses.

This subsection presents key steps of processing network traffic and ensuring security in blockchain system. These steps describe how network nodes interact for detecting anomalies, data verification, and automatic blocking of suspicious IP addresses. Every step plays an important rope in creating a reliable and efficient system of protecting from cyberattacks, ensuring data integrity and rapid response to threats.

The steps of processing network traffic and ensuring security in blockchain system are the following:

Step 1. Formation of data block.

Each blockchain network node processes the entering network traffic and, if there is an anomaly or suspicious activity (e.g., multi-vector attack), forms data block *B*. This block includes the following elements:

$$B = \{\text{Block ID, Data, Previous Hash, Timestamp, Signature}\},$$

where Block ID is the unique block identifier, Data is the information on traffic and possible attacks (e.g., IP addresses, type of attack, and timestamps), Previous Hash is the hash of the previous block in blockchain for maintaining continuous chain, Timestamp is the block formation time, and Signature is the digital signature of the node that formed the block.

Step 2. PoV consensus mechanism.

After the block is formed, it is transferred to other network nodes for verification using the PoV consensus mechanism.

The block is validated by voting of network nodes in the following order:

- each node analyzes the data block B, tests its integrity and reliability, and then sends its voice V_i (voting may be binary: $V_i = 1$ for acceptance of block and $V_i = 0$ for rejection of block);
- the probability that the block is valid is calculated as

$$P_{\text{valid}} = \frac{\sum_{i=1}^{N} V_i}{N}.$$

If $P_{\text{valid}} \geq 0.5$ (most nodes support the block), then the block is considered valid and is added to the distributed ledger, and if $P_{\text{valid}} < 0.5$, then the block is rejected.

Step 3. Refreshing of distributed ledger

After reaching consensus and supporting block B, it is added to the distributed ledger. Each item in the distributed ledger is related to the previous block through the Previous Hash. Which ensures continuous and invariable data chain. A new block is added to the distributed ledger:

$$B_{\text{new}} = \{\text{Hash}(B_{\text{prev}}), \text{Data}_{\text{new}}, \text{Timestamp}_{\text{new}}, \text{Signature}_{\text{new}}\},$$

where $\text{Hash}(B_{\text{prev}})$ is the hash of the previous block, which guarantees the integrity of the entire chain.

Step 4. Using action modules for automatic blocking of IP addresses.

Blockchain system uses action modules for automatic actions when detecting attack. Action modules Action modules automatically block IP addresses, send notice, and refresh blacklists in network. The structure of action module can have the following form:

$$\text{AM} = \{\text{Source IP (SIP)}, \text{Destination IP (DIP)}, \text{Signature}, \text{Blacklisted IP}, \text{Attack Label}\},$$

where Source IP (SIP) is the IP address, from which traffic enters; Destination IP (DIP) is the IP address of target device; Signature is the digital signature of data for information authentication; Blacklisted IP is the list of IP addresses, which were blocked after detecting attack; and Attack Label is the attack label, which contains type of attack (e.g., DDoS, SQL injection, multi-vector attack).

Step 5. IP address blocking.

Once the data block on attack is confirmed and added to blockchain, action module automatically blocks harmful IP addresses in network. For example, if DDoS traffic is detected, then the IP address of the SIP attacking device is added to the Blacklisted IP blacklist through action module

$$\text{AM(SIP)} = \text{Blacklisted IP}.$$

These data are refreshed at every network node through distributed blockchain structure, which guarantees the consistency of actions of all participants.

Step 6. Block validation time.

For each block B, the block validation time T_B is calculated, which depends on the time t_{vote} of voting of all nodes, the time t_{contract} of performance of all action modules, and the time t_{transmit} of block transmission between nodes:

$$T_B = t_{\text{vote}} + t_{\text{contract}} + t_{\text{transmit}}.$$

The optimization of block validation time is critical for IoT networks with limited resources and high data exchange rate.

2.4. Pruning

Pruning is used to reduce computational costs and optimize model operation at low-power IoT devices. After model learning, minor neurons and their connections are pruned to reduce model volume without significant impairment of its accuracy.

Pruning is performed as follows:

- after learning of neural network, the weights of its connections are analyzed. If the weights are below a given level, then the connections are pruned;
- the model is restarted with a reduced number of neurons and parameters, which reduces its computational complexity and memory requirements.

The main idea is to prune unnecessary or minor neurons or change weights after model learning, insignificantly reducing its productivity.

The main elements and variables are the following:

$\mathbf{W}$—weight matrix of neural network;

$\mathbf{b}$—bias vector of neurons;

$f(\mathbf{W})$—activation function for network weights;

θ—threshold for weight pruning;

$\mathbf{M}$—masking matrix for weight pruning;

n_{total}—total number of parameters (weights) in neural network;

n_{pruned}—number of pruned weights;

p—fraction of pruned weights or neurons.

2.4.1. Determination of significance of weights and neurons

After the neural network is learned, it is necessary to determine, which weights $\mathbf{W}$ in neural network affect least on output values and can be pruned. This is done by calculating the significance of each weight $W_{i,j}$. As a measure of significance, the metric of absolute value of weight can be used, in which the smaller the weight, the less significant this weight for neuron activation:

$$\text{Significance}(W_{i,j}) = |W_{i,j}|. \tag{11}$$

If the weight is close to zero, then its influence on network output is minimal, and such a weight can be pruned.

2.4.2. Pruning threshold

To determine, which weights should be pruned, threshold θ is introduced. The weights the absolute value of which is smaller than θ are considered minor and removed (equated to zero 0):

$$W_{i,j} = 0 \text{ if } |W_{i,j}| < \theta. \tag{12}$$

The threshold θ is chosen empirically or optimized during experiments. This threshold can be static or dynamic, adapted by analyzing model structure.

2.4.3. Masking matrix

Lest weight pruning affect neurons that have a significant effect on the output values of the model and its productivity, masking matrix $\mathbf{M}$ is used, denoting which weights should be retained, and which should be zeroed:

$$M_{i,j} = \begin{cases} 1, \text{ if } |W_{i,j}| \geq \theta, \\ 0, \text{ if } |W_{i,j}| < \theta. \end{cases} \tag{13}$$

Pruned masking matrix:

$$\mathbf{W}_{\text{pruned}} = \mathbf{W} \odot \mathbf{M}, \tag{14}$$

where $\odot$ is the elementwise product of the weight matrix $\mathbf{W}$ and the masking matrix $\mathbf{M}$. This guarantees that only significant weights participate in computations, and minor weights are pruned.

2.4.4. Estimation of pruned weight fraction

The fraction of pruned weights or neurons is calculated as

$$p = \frac{n_{\text{pruned}}}{n_{\text{total}}}, \tag{15}$$

where n_{pruned} is the number of pruned weights, i.e., weights for which $|W_{i,j}| < \theta$; and n_{total} is the total number of weights in the model.

2.4.5. Iterative pruning

Simple pruning threshold may be insufficiently efficient for all network layers, especially for deep models with numerous layers. Therefore, iterative pruning may be used: weights are pruned not at once but stepwise, with a gradual increase in threshold θ.

At each iteration, the weights are recalculated using the masking matrix:

$$\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} \odot \mathbf{M}. \tag{16}$$

Then the network is relearned on new data to restore its accuracy after pruning. This process is repeated several times until the pruned weight fraction reaches the desired level p.

2.4.6. The main steps of the developed simulation model and metric for estimating the model quality after weight pruning

The main steps of the model operation are the following:

(1) traffic analysis using CNN + LSTM for detecting spatiotemporal signs and identifying anomalies;

(2) model optimization: the Mayfly algorithm automatically tunes model hyperparameters depending on the conditions of network and data, which ensures its adaptability;

(3) suspicious IP addresses and traffic are verified and blocked through blockchain consortium. If the attack on IP addresses is confirmed, they are blocked through action modules;

(4) neuron pruning: after the initial learning and verification of the model, neurons are pruned to reduce computational complexity and model adaptation to resources of IoT devices.

After pruning, it is important to estimate the changes in model productivity and resource intensity. The main metrics of model quality after weight pruning are the following:

(1) the fraction of accurate predictions among the total amount of predictions (accuracy of correct predictions):

$$E_1 = \text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{17}$$

where TP is the number of true positive predictions (correct predictions of attacks), TN is the number of true negative predictions (correct predictions of normal traffic), FP is the number of false positive predictions (false alerts), and FN is the number of false negative predictions (lost attacks);

(2) recall:

$$E_2 = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{18}$$

which determines the model ability to detect all attacks in a sample;

(3) precision (fraction of true positive predictions among all positive predictions (accuracy of true positive predictions)):

$$E_3 = \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{19}$$

which determined how much model predictions of positive classes are correct;

(4) F1 metric:

$$E_4 = 2\frac{E_2 E_3}{E_2 + E_3}, \tag{20}$$

which is a harmonic mean between recall and precision;

(5) time of data processing and performing model predictions:

$$E_5 = T_{\text{pruned}} = T_{\text{original}}(1 - p), \tag{21}$$

where T_{pruned} is the computational time after pruning, T_{original} is the computational time by the initial model, and p is the fraction of pruned neurons;

(6) memory usage–(reduction of memory usage after weight pruning):

$$E_6 = M_{\text{pruned}} = M_{\text{original}}(1 - p), \tag{22}$$

where M_{pruned} is the memory required for storing pruned model, and M_{original} is the memory necessary for storing the initial model.

3. IMPLEMENTATION AND EXPERIMENT

3.1. Design of experiment

In this section, we experimentally tested the developed simulation model of detecting multi-vector attacks taking into account the limitations of computational and informational resources of IoT devices.

The experiment uses CIC IoT Dataset 2023, which contains data of network traffics, both normal, and attacking, with various characteristics and signs, such as:

- IP addresses (Source/Destination);
- ports (Source/Destination);
- connection time;
- packet size;
- protocols (TCP[12], UDP[13], HTTP, DNS);
- attack mark (e.g., DDoS, SQL injection, Brute Force).

[12] Transmission Control Protocol.

[13] User Datagram Protocol.

The dataset is divided into several classes:

- normal traffic;
- attacking traffic (various types of attacks).

The experiment compares the results of the developed simulation model with several similar methods used to detect attacks on IoT networks.

The CIC IoT Dataset 2023 data were normalized using Min-Max Scaling to reduce all signs to the range [0, 1]. Outliers were removed by Interquartile Range (IQR) method, and new signs were generated by aggregation of time characteristics of traffic (e.g., mean number of packets in 10 s).

The testing was performed on a workstation based on Macbook Pro notebook (Apple Inc., USA) with M2 Pro processor (includes 12 processor cores (8 performance cores and 4 efficiency cores), 19 graphics engines, and 16-core neural coprocessor) and16-GB random access memory with a throughput efficiency of about 200 GB/s.

3.2. Structure of experiment

The experiment was performed in several steps:

(1) Data preparation, at which the CIC IoT Dataset 2023 data are divided into learning and testing samplings in the ratio 70 : 30. Then, the data are preliminarily processed: signs are normalized, outliers are removed, and new signs are generated (if necessary).

(2) Model learning, at which the proposed model is learned based on the hydride architecture CNN + LSTM/GRU using neuron pruning to reduce computational costs.

The model hyperparameters are optimized using the Mayfly algorithm.

For comparison, other models are also learned, such as Random Forest, SVM, Deep Learning (MLP).

The processing time of a single data packet is found as the sum of convolution time (1), LSTM/GRU processing time (2)–(9), and classification time (10). Owing to neuron pruning (11)–(16), the model significantly reduces the number of parameters, which decreases the computational costs and improves productivity on devices with limited resources. This makes it possible to efficiently use the model under real conditions, such as real-time monitoring systems, where fast response and minimum memory consumption are important. The quality of the simulation model is estimated using metrics (17)–(22).

3.3. Analysis of the obtained results

The efficiency of the proposed model was estimated in experimental studies. Table estimates the efficiency using evaluation metrics (17)–(22) of various models, including the proposed model based on the CNN + LSTM/GRU hybrid architecture using neuron pruning.

Table. Experimental results

Model	Evaluation metrics					
	E_1, %	E_2, %	E_3, %	E_4, %	E_5, ms	E_6, MB
Random Forest	96.5	95.7	97.1	96.4	35	220
SVM	94.3	92.6	94.5	93.5	50	250
Deep Learning (MLP)	97.8	97.2	98.0	97.6	20	210
Simulation model CNN + LSTM/GRU	99.1	99.3	98.9	99.1	12	180

Figure 1 compares the results of operation of the proposed simulation model and its analogs in accordance with evaluation metrics (17)–(20). Figure 1 shows that the CNN + LSTM/GRU model significantly exceeds the other models in all presented metrics, which confirms its high efficiency in detecting multi-vector attacks.

Figure 2 compares the results of operation of the proposed simulation model and its analogs in accordance with evaluation metrics E_5 (21). Figure 2 shows that the model CNN + LSTM/GRU has the minimum data processing time and the minimum prediction performance time (12 ms), which makes it particularly suitable for real-time use, whereas the other models require much more time.

Figure 3 compares the results of operation of the proposed simulation model and its analogs in accordance with quality metric E_6 (22). Figure 5 shows that the model CNN + LSTM/GRU requires the minimum memory volume (180 MB) for processing the input data of 1 million examples with 10 signs, which makes it more efficient for using on devices with limited computational resources in comparison with the other models, such as Random Forest and SVM.

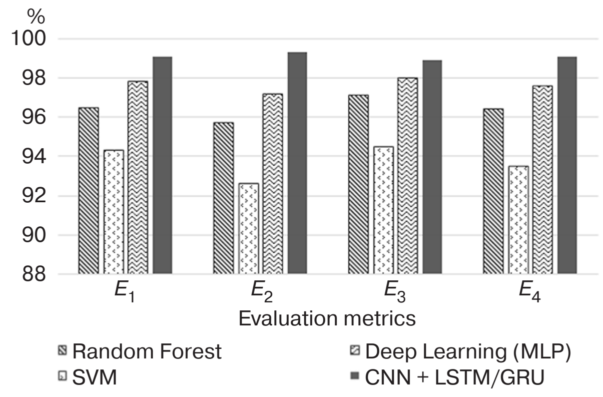

Fig. 1. Comparison of the results of operation of the developed simulation model CNN + LSTM/GRU and its analogues in accordance with metrics E_1 (17) – E_4 (20)

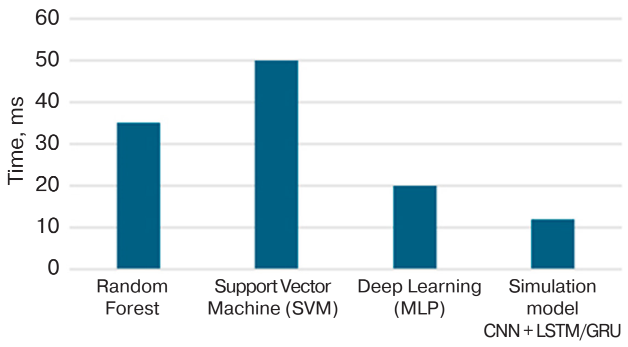

Fig. 2. Comparison of the results of operation of the developed simulation model CNN + LSTM/GRU and its analogues in accordance with metric E_5 (21)

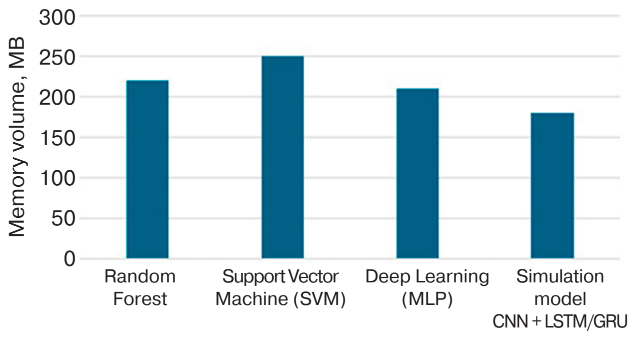

Fig. 3. Comparison of the results of operation of the developed simulation model CNN + LSTM/GRU and its analogues in accordance with metric E_6 (22)

The performed experiment confirms the following:

(1) The developed simulation model attained high accuracy of detecting attacks at a level of 99.1%, which suggests its ability to efficiently identify both known and new types of attacks in real time. Thus, the proposed architecture based on the CNN + LSTM/GRU hybrid model can be successfully applied in the context of analyzing network traffic.

(2) The F1 metric of the developed simulation model is 99.1%, which indicates high balance between precision (19) and recall (18). This means that the model not only accurately identifies attacks, but also minimizes the number of false alarms and miss.

(3) The request processing time in the proposed simulation model at hardware resources stated in 3.1 of this work is reduced to 12 ms, making this model particularly useful for systems requiring fast response, such as real-time monitoring systems. This represents a significant advantage over the other models, which require more processing time.

(4) The developed simulation model uses only 180 MB memory, making it suitable for implementing on devices with limited computational resources. This is particularly important for IoT devices, which are often limited in memory and computational power.

(5) Neuron pruning significantly reduced the number of model parameters from 1.5 million to 300 thousand, which, in turn, decreased computational costs by 80% and improved productivity. This confirms that model optimization approaches contribute significantly to its successful use under limited resources.

CONCLUSIONS

In this work, we have proposed a simulation model of a scalable method for detecting multi-vector attacks on IoT devices that takes into account the limitations of computational and informational resources. The creation of an efficient solution capable of detecting attacks with high accuracy is a key objective given the growing security threats in IoT.

The proposed model is based on a hybrid architecture of neural networks that combines convolutional neural networks CNN for analyzing spatial dependencies and long short-term memory networks LSTM for analyzing time dependencies of network traffic. An important aspect is pruning, which significantly reduces the number of model parameters to decrease computational costs. The use of blockchain technologies with a PoV consensus mechanism ensures data security and decentralized verification, which is critically important for protecting IoT networks from multi-vector attacks.

The experimental testing using the CIC IoT Dataset 2023 dataset demonstrated the high efficiency of the proposed model. The achieved attack detection accuracy 99.1% confirms its ability to exactly identify both known and new types of attacks in real time. The F1 metric of 99.1% indicates a balance between precision and recall, which is critically important for cybersecurity systems in which both false alarms and unidentified attacks should be minimized. In addition to high accuracy, the request processing time was reduced to 12 ms. This allows the model to function efficiently under rapid response conditions such as real-time monitoring systems. Memory use was also optimized to only 180 MB, which makes it suitable for devices with limited computational resources.

Thus, the developed simulation model exceeds the existing solutions in key metrics, such as precision, processing time, and memory use. The high efficiency of the model during multi-vector threats to IoT is ensured by its hybrid architecture, neuron pruning, and decentralized verification.

This work opens new horizons for further research in cybersecurity that will lead to efficient solutions for protecting IoT networks from complex cyberthreats. Future studies should aim to integrate additional machine learning and deep learning methods for increasing the accuracy and stability of the model to new types of attacks. It is also worth considering the possibility of optimizing algorithms to reduce computational costs and increase data processing rates. Under current conditions of increased device numbers and traffic volumes, it becomes crucial to continue to improve the scalability and stability of blockchain-oriented solutions.

ACKNOWLEDGMENTS

This research was supported by the Information Security grant from the Moscow Technical University of Communications and Informatics (MTUSI) (agreement No. 40469/17-23-K, "Development of an intrusion detection method using multi-vector attack scenarios in a decentralized IoT environment."

Authors' contributions

V.I. Petrenko—research idea, planning the study, and scientific editing the article.

F.B. Tebueva—research idea, planning the study, and scientific editing the article.

M.G. Ogur—conducting the research, performing the experimental part of the work, analysis of the obtained data, formulating the results, and writing the text of the article.

G.I. Linets—consultations on conducting the research and scientific editing the article.

V.P. Mochalov—consultations on conducting the research and scientific editing the article.

REFERENCES

1. Sen Ö., Ivanov B., Henze M., Ulbig A. Investigation of Multi-stage Attacks and Defense Modeling for Data Synthesis. In: *Proceedings of the International Conference on Smart Energy Systems and Technologies* (*SEST*). IEEE; 2023. P. 1–12. https://doi.org/10.1109/SEST57387.2023.10257329
2. Lysenko S., Bobrovnikova K., Kharchenko V., Savenko O. IoT Multi-Vector Cyberattack Detection Based on Machine Learning Algorithms: Traffic Features Analysis, Experiments, and Efficiency. *Algorithms*. 2022;15(7):239. https://doi.org/10.3390/a15070239
3. Aguru A., Erukala S. OTI-IoT: A Blockchain-based Operational Threat Intelligence Framework for Multi-vector DDoS Attacks. *ACM Trans. Internet Technol*. 2024;24(3):15.1–15.31. https://doi.org/10.1145/3664287
4. Ipole-Adelaiye N., Tatama F.B., Egena O., Jenom M., Ibrahim L. Detecting Multi-Vector Attack Threats Using Multilayer Perceptron Network. *IRE Journals*. 2024;8(1):119–123.
5. Pakmehr A., Aßmuth A., Taheri N., Ghaffari A. DDoS attack detection techniques in IoT networks: a survey. *Cluster Comput*. 2024;27(4):14637–14668. https://doi.org/10.1007/s10586-024-04662-6
6. Alhakami W. Evaluating modern intrusion detection methods in the face of Gen V multi-vector attacks with fuzzy AHP-TOPSIS. *PLoS One*. 2024;19(5):e0302559. https://doi.org/10.1371/journal.pone.0302559
7. Saiyed M.F., Al-Anbagi I. Deep Ensemble Learning With Pruning for DDoS Attack Detection in IoT Networks. *IEEE Trans. Machine Learning Commun. Networks*. 2024;2:596–616. https://doi.org/10.1109/TMLCN.2024.3395419
8. Liebl S. *Threat Modelling for Internet of Things Devices*. Research Report 2023 of the Technical University OTH Amberg-Weiden. 2023. Available from URL: https://www.researchgate.net/publication/369488078. Accessed February 25, 2025.
9. Aguru A.D., Erukala S.B. A lightweight multi-vector DDoS detection framework for IoT-enabled mobile health informatics systems using deep learning. *Inf. Sci*. 2024;662:120209. https://doi.org/10.1016/j.ins.2024.120209
10. Petrenko V.I., Tebueva F.B., Ogur M.G., Linets G.I., Mochalov V.P. Methodology for detecting and countering multi-vector threats to information security of a decentralized IoT system. *Int. J. Open Inf. Technol*. 2025;13(1):13–24 (in Russ.).
11. Leng S., Guo Y., Zhang L., Hao F., Cao X., Li F., Kou W. Online and Collaboratively Mitigating Multi-Vector DDoS Attacks for Cloud-Edge Computing. In: *ICC 2024 – International Conference on Communications*. 2024. P. 1394–1399. https://doi.org/10.1109/ICC51166.2024.10623052
12. Ali M., Saleem Y., Hina S., Shah G.A. DDoSViT: IoT DDoS attack detection for fortifying firmware Over-The-Air (OTA) updates using vision transformer. *Internet of Things*. 2025;30:101527. https://doi.org/10.1016/j.iot.2025.101527
13. Dalal S., Lilhore U.K., Faujdar N., Simaiya S., et al. Next-generation cyberattack prediction for IoT systems: leveraging multi-class SVM and optimized CHAID decision tree. *J. Cloud Comput*. 2023;12:137. https://doi.org/10.1186/s13677-023-00517-4

14. Zahid F., Funchal G., Melo V., Kuo M.M.Y., et al. DDoS attacks on smart manufacturing systems: A cross-domain taxonomy and attack vectors. In: *2022 20th IEEE International Conference on Industrial Informatics* (*INDIN*). 2022. P. 214–219. https://doi.org/10.1109/INDIN51773.2022.9976172
15. Lungu N., Dash B.B., De U.C., Dash B.B., et al. Multi-vector Monitoring, Detecting and Classifying GPU Side-Channel Attack Vectors on a Secure GPU Execution Framework. In: *2024 8th International Conference on I-SMAC* (*IoT in Social, Mobile, Analytics and Cloud*). 2024. P. 500–505. https://doi.org/10.1109/I-SMAC61858.2024.10714895

About the Authors

Vyacheslav I. Petrenko, Cand. Sci. (Eng.), Associate Professor, Head of the Department of Organization and Technology of Information Security, Prof. Nikolay Chervyakov Faculty of Mathematics and Computer Sciences, North-Caucasus Federal University (1, Pushkina ul., Stavropol, 355017 Russia). E-mail: vipetrenko@ncfu.ru. Scopus Author ID 57189512011, ResearcherID A-3196-2017, RSCI SPIN-code 3923-4295, https://orcid.org/0000-0003-4293-7013

Fariza B. Tebueva, Dr. Sci. (Phys.-Math.), Associate Professor, Professor, Department of Computational Mathematics and Cybernetics, Prof. Nikolay Chervyakov Faculty of Mathematics and Computer Sciences, North-Caucasus Federal University (1, Pushkina ul., Stavropol, 355017 Russia). E-mail: ftebueva@ncfu.ru. Scopus Author ID 57189512319, ResearcherID H-4548-2017, RSCI SPIN-code 9343-7504, https://orcid.org/0000-0002-7373-4692

Maxim G. Ogur, Senior Lecturer, Department of Computational Mathematics and Cybernetics, Prof. Nikolay Chervyakov Faculty of Mathematics and Computer Sciences, North-Caucasus Federal University (1, Pushkina ul., Stavropol, 355017 Russia). E-mail: ogur26@gmail.com. ResearcherID B-1332-2017, RSCI SPIN-code 7180-6971, https://orcid.org/0000-0002-2387-0901

Gennady I. Linets, Dr. Sci. (Eng.), Professor, Department of Digital, Robotic Systems and Electronics, Institute of Advanced Engineering, North-Caucasus Federal University (1, Pushkina ul., Stavropol, 355017 Russia). E-mail: kbytw@mail.ru. Scopus Author ID 6506372022, RSCI SPIN-code 1452-6823, https://orcid.org/0000-0002-2279-3887

Valery P. Mochalov, Dr. Sci. (Eng.), Professor, Department of Digital, Robotic Systems and Electronics, Institute of Advanced Engineering, North-Caucasus Federal University (1, Pushkina ul., Stavropol, 355017 Russia). E-mail: mochalov.valery2015@yandex.ru. Scopus Author ID 57202300745, RSCI SPIN-code 8695-1648, https://orcid.org/0000-0002-5131-5649

Об авторах

Петренко Вячеслав Иванович, к.т.н., доцент, заведующий кафедрой организации и технологии защиты информации, факультет математики и компьютерных наук имени профессора Н.И. Червякова, ФГАОУ ВО «Северо-Кавказский федеральный университет» (355017, Россия, Ставрополь, ул. Пушкина, д. 1). E-mail: vipetrenko@ncfu.ru. Scopus Author ID 57189512011, ResearcherID A-3196-2017, SPIN-код РИНЦ 3923-4295, https://orcid.org/0000-0003-4293-7013

Тебуева Фариза Биляловна, д.ф.-м.н., доцент, профессор кафедры вычислительной математики и кибернетики, факультет математики и компьютерных наук имени профессора Н.И. Червякова, ФГАОУ ВО «Северо-Кавказский федеральный университет» (355017, Россия, Ставрополь, ул. Пушкина, д. 1). E-mail: ftebueva@ncfu.ru. Scopus Author ID 57189512319, ResearcherID H-4548-2017, SPIN-код РИНЦ 9343-7504, https://orcid.org/0000-0002-7373-4692

Огур Максим Геннадьевич, старший преподаватель, кафедра вычислительной математики и кибернетики, факультет математики и компьютерных наук имени профессора Н.И. Червякова, ФГАОУ ВО «Северо-Кавказский федеральный университет» (355017, Россия, Ставрополь, ул. Пушкина, д. 1). E-mail: ogur26@gmail.com. ResearcherID B-1332-2017, SPIN-код РИНЦ 7180-6971, https://orcid.org/0000-0002-2387-0901

Линец Геннадий Иванович, д.т.н., профессор, профессор департамента цифровых, робототехнических систем и электроники, институт перспективной инженерии, ФГАОУ ВО «Северо-Кавказский федеральный университет» (355017, Россия, Ставрополь, ул. Пушкина, д. 1). E-mail: kbytw@mail.ru. Scopus Author ID 6506372022, SPIN-код РИНЦ 1452-6823, https://orcid.org/0000-0002-2279-3887

Мочалов Валерий Петрович, д.т.н., профессор, профессор департамента цифровых, робототехнических систем и электроники, институт перспективной инженерии, ФГАОУ ВО «Северо-Кавказский федеральный университет» (355017, Россия, Ставрополь, ул. Пушкина, д. 1). E-mail: mochalov.valery2015@yandex.ru. Scopus Author ID 57202300745, SPIN-код РИНЦ 8695-1648, https://orcid.org/0000-0002-5131-5649

Translated from Russian into English by Vladislav Glyanchenko
Edited for English language and spelling by Thomas A. Beavitt