ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2024.0469

ADVANCES IN NETWORK SECURITY FOR A RESILIENT DIGITAL CYBERSPACE INFRASTRUCTURE

Rajender Udutha[1], Sunil Kumar[2], Allanki Sanyasi Rao[3] and Srinivasa Rao Dhanikonda[4]
[1]*Department of Electronics and Communication Engineering, Vaageswari College of Engineering, India*
[2]*School of Information Technology, Auro University, India*
[3]*Department of Electronics and Communication Engineering, Christu Jyothi Institute of Technology and Science, India*
[4]*Department of Information Technology, BVRIT Hyderabad College of Engineering for Women, India*

Abstract

In network security, safeguarding digital cyberspace infrastructure against malicious intrusions remains a paramount concern. This study addresses this imperative by proposing a novel approach that integrates digital watermarking, Huffman coding, and sophisticated attack classification techniques within the Controller Area Network (CAN) protocol framework. The proliferation of interconnected systems and the increasing sophistication of cyber threats necessitate novel strategies to fortify network defenses and ensure resilience in the face of adversarial activities. The research problem revolves around the need to develop an effective method for detecting and categorizing reconnaissance and violent attacks within cyberspace networks, particularly those utilizing the CAN protocol. Despite advancements in security protocols and intrusion detection systems, existing methodologies often fall short in accurately distinguishing between benign and malicious network activities, leaving critical infrastructure vulnerable to exploitation. This research uses digital watermarking to embed metadata within network packets, Huffman coding for efficient data compression, and advanced attack classification algorithms for real-time threat identification. The experimental results demonstrate the efficacy of the proposed approach in accurately differentiating between reconnaissance probes and violent attacks, thereby enabling timely and targeted responses to mitigate security threats.

Keywords:

Network Security, Controller Area Network (CAN) Protocol, Digital Watermarking, Huffman Coding, Attack Classification

1. INTRODUCTION

In digital landscape, ensuring the security and resilience of cyberspace infrastructure is of paramount importance. With the proliferation of networked systems spanning various sectors such as automotive, industrial control, and IoT devices, the need for robust security measures has become increasingly pressing [1]. Among the myriad challenges facing network security, the Controller Area Network (CAN) protocol, commonly utilized in vehicular and industrial environments, presents unique vulnerabilities that demand novel solutions to safeguard against malicious intrusions [2].

The CAN protocol, originally designed for reliable real-time communication in automotive applications, has since found widespread adoption in diverse domains due to its efficiency and simplicity [3]. However, its inherent lack of built-in security features renders CAN networks susceptible to various cyber threats, including unauthorized access, data manipulation, and denial-of-service attacks [4]. As CAN-based systems continue to proliferate, securing these networks against malicious exploitation has emerged as a critical imperative [5].

Securing CAN networks poses several significant challenges. Firstly, the decentralized nature of CAN architecture, characterized by multiple nodes communicating over a shared bus, complicates the implementation of robust security measures [6]. Additionally, the resource-constrained nature of many CAN-enabled devices limits the feasibility of deploying complex security solutions. Furthermore, the dynamic and heterogeneous nature of modern cyber threats necessitates adaptive and proactive defense mechanisms capable of identifying and mitigating emerging attack vectors in real-time [7].

The primary focus of this research is to address the challenge of enhancing network security within CAN-based cyberspace infrastructure. Specifically, the research aims to develop a comprehensive methodology that effectively detects and classifies reconnaissance and violent attacks targeting CAN networks. Reconnaissance attacks involve probing the network to gather intelligence and identify potential vulnerabilities, while violent attacks are aimed at disrupting network operations or causing damage.

The objectives of this research are twofold. Firstly, to devise a robust methodology that integrates digital watermarking, Huffman coding, and advanced attack classification algorithms within the CAN protocol framework to enhance network security. Secondly, to evaluate the efficacy of the proposed approach through extensive experimentation and validation in real-world scenarios.

The novelty of this research lies in its approach to addressing the security challenges inherent in CAN-based networks. By combining digital watermarking techniques to embed metadata within network packets, Huffman coding for efficient data compression, and sophisticated attack classification algorithms, the proposed method offers a solution for detecting and categorizing malicious activities in real-time.

2. RELATED WORKS

This seminal work explores the vulnerabilities of the CAN protocol and demonstrates various attacks, including message injection and bus-off attacks, highlighting the critical need for improved security mechanisms in CAN-based systems.

While focusing on wireless sensor networks (WSNs), this survey discusses intrusion detection techniques applicable to CAN networks, such as anomaly detection and signature-based methods, providing insights into potential strategies for securing CAN-based systems.

An intrusion detection system (IDS) tailored in [8] for CAN networks, employing anomaly detection algorithms to identify

abnormal behavior and potential security breaches. The study emphasizes the importance of real-time monitoring and response in mitigating cyber threats in CAN environments.

A review in [9] surveys existing security mechanisms and protocols aimed at enhancing the security of CAN networks. It provides an overview of encryption techniques, authentication protocols, and intrusion detection systems relevant to mitigating security risks in CAN-based systems.

While focusing on smart grid communications, [10] discusses cybersecurity challenges applicable to CAN networks, such as data integrity, confidentiality, and authentication. It highlights the importance of implementing robust security measures to protect critical infrastructure against cyber threats.

A machine learning-based approach in [11] for real-time anomaly detection in CAN bus networks. By leveraging Long Short-Term Memory (LSTM) networks, the model can effectively identify abnormal patterns indicative of cyber-attacks, contributing to the development of proactive security measures for CAN-based systems.

These works [12] collectively contribute to the body of knowledge surrounding security challenges and solutions in CAN-based networks, providing valuable insights and methodologies for enhancing the resilience of cyberspace infrastructure against malicious intrusions.

3. PROPOSED METHOD

The proposed method aims to enhance the security of cyberspace infrastructure utilizing the Controller Area Network (CAN) protocol by integrating digital watermarking, Huffman coding, and advanced attack classification techniques.

Digital watermarking in Fig.1 involves embedding imperceptible information, or metadata, into digital content. In the context of network security, this technique can be applied to network packets transmitted over the CAN protocol. Each packet is embedded with a digital watermark containing metadata such as source, destination, and integrity verification codes. This embedding process is imperceptible to human senses but can be detected and decoded by specialized algorithms.

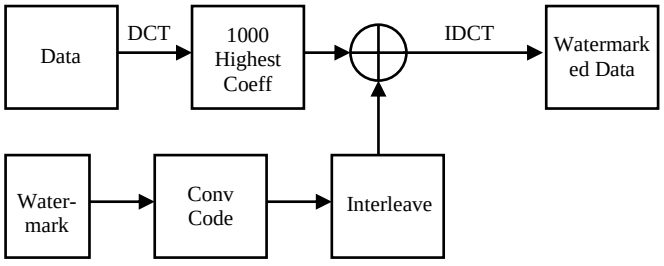
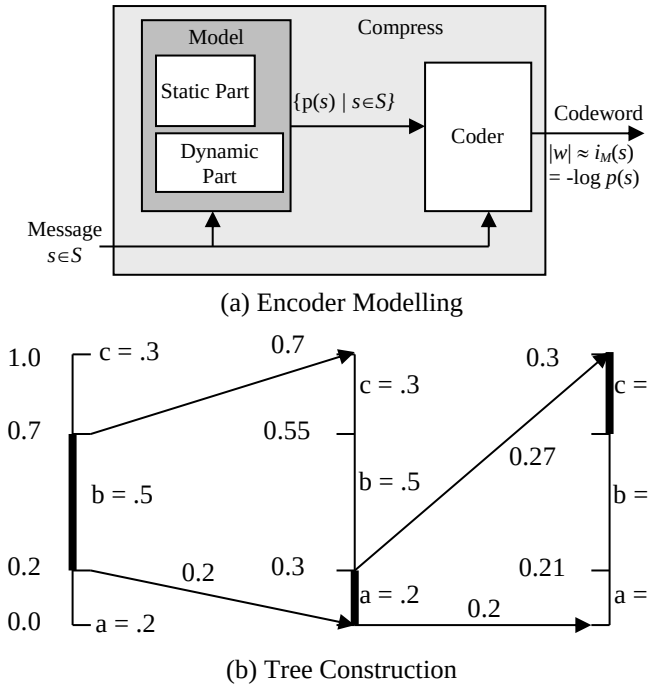


Fig.1. Digital Watermarking

Huffman coding in Fig.2 is a method for lossless data compression, where variable-length codes are assigned to input characters based on their frequency of occurrence, with more frequent characters assigned shorter codes. In the proposed method, Huffman coding is applied to compress the payload of each network packet before transmission. This compression helps reduce bandwidth consumption and optimize network performance.



(a) Encoder Modelling

(b) Tree Construction

Fig.2. Huffman Coding

The proposed method employs sophisticated attack classification algorithms to analyze network traffic patterns and characteristics in real-time. These algorithms are designed to differentiate between reconnaissance probes and violent attacks targeting the CAN network. By examining packet headers, payload contents, and other relevant features, the system can accurately classify incoming packets as either benign or malicious.

Network packets are generated by various nodes within the Controller Area Network (CAN) infrastructure. Each packet is processed to embed a digital watermark containing metadata such as source, destination, and integrity verification codes. The embedding process ensures that the watermark is imperceptible to human senses but can be detected by specialized algorithms. The payload of each packet is compressed using Huffman coding. Huffman coding assigns variable-length codes to input characters based on their frequency of occurrence, optimizing compression efficiency. The digitally watermarked and Huffman coded packets are transmitted over the CAN protocol network. Upon reception, each packet is processed to extract the embedded metadata and verify its integrity. The payload of the packet is decompressed using Huffman decoding, restoring the original data. The decompressed packet undergoes analysis using advanced attack classification algorithms. These algorithms examine packet headers, payload contents, and other relevant features to determine the presence of malicious activities. Based on the results of the analysis, the packet is classified as either benign or malicious. Reconnaissance probes and violent attacks targeting the CAN network are identified and distinguished.

3.1 PERFORMANCE ANALYSIS

- **Detection Accuracy:** Measures the proportion of correctly identified malicious packets among all packets analyzed. High detection accuracy indicates the effectiveness of the proposed method in identifying threats accurately.

- **False Positive Rate:** Represents the ratio of benign packets incorrectly classified as malicious. A low false positive rate indicates that the system minimizes unnecessary alerts and false alarms.
- **Response Time:** Quantifies the time taken by the system to detect and respond to security threats. Lower response times indicate quicker mitigation of attacks and improved network resilience.
- **Network Throughput:** Measures the rate at which packets are successfully transmitted and processed by the network. The proposed method should maintain high network throughput while ensuring robust security measures.
- **Resource Utilization:** Evaluates the computational resources and memory required by the system to implement the proposed security measures. Optimal resource utilization ensures efficient operation without excessive overhead.

4. DIGITAL WATERMARKING

Digital watermarking involves embedding imperceptible information into digital content. In network packets, this information is embedded within the packet data. Let P represent the original packet payload, and W represent the digital watermark information to be embedded. The process of embedding the watermark can be represented by:

$$P_w = P+W \quad (1)$$

where P_w is the watermarked packet payload obtained by combining the original packet payload P with the watermark information W.

the process of embedding the watermark involves modifying certain bits or bytes of the packet payload in a way that is imperceptible to human senses but detectable by specialized algorithms. This modification can be achieved using various techniques such as LSB (Least Significant Bit) embedding, spread spectrum modulation, or frequency domain techniques.

Algorithm: Digital Watermarking for Network Packets

Input: P: Original packet payload; W: Digital watermark information; α: Embedding strength parameter

Output: P_w: Watermarked packet payload

1) $P_w \leftarrow P$
2) Find the length L of the watermark information W.
3) Divide the packet payload P into L segments or blocks $\{p_1,p_2,...,p_L\}$.
4) For each segment p_i:
   i) Calculate the embedding strength
   $$\beta_i = \alpha \times p_i. \quad (2)$$
   ii) Modify the segment p_i by adding the corresponding watermark information w_i scaled by
   $$\beta_i: p_i' = p_i + \beta_i \times w_i \quad (3)$$
   iii) Update the watermarked packet payload: $P_w = \{p_1', p_2', ..., p_L'\}$

where,

P: Original packet payload

W: Digital watermark information

P_w: Watermarked packet payload

α: Embedding strength parameter (a scalar value)

β_i: Embedding strength for segment p_i

p_i: Segment or block of the original packet payload

p_i': Modified segment or block of the watermarked packet payload

w_i: Watermark information for segment pi

L: Length of the watermark information W

The embedding strength parameter α controls the intensity of the embedding process, influencing the visibility and robustness of the watermark. The watermark information W can be binary, textual, or any form of data desired to be embedded within the packet payload.

5. HUFFMAN CODING COMPRESSION

Huffman coding is a technique used for lossless data compression, where variable-length codes are assigned to input characters based on their frequency of occurrence.

- *Frequency Calculation*: For each unique symbol s_i in the input data: $f(s_i)$ represents the frequency of occurrence of symbol s_i
- *Construction of Huffman Tree*: Construct a frequency table F containing the frequencies of all unique symbols. Build a Huffman tree based on the frequency table F.
- *Generation of Huffman Codes*: For each symbol s_i:
  $$s_i = C(s_i) \quad (4)$$

where, $C(s_i)$ represents the Huffman code assigned to symbol s_i.

- *Compression*: For each symbol s_i in the input data, replace it with its corresponding Huffman code $C(s_i)$ to compress the data.

Consider a set of symbols $\{s_1,s_2,...,s_n\}$ with corresponding frequencies $\{f(s_1),f(s_2),...,f(s_n)\}$. After constructing the Huffman tree and generating Huffman codes, the compressed data can be represented as a sequence of Huffman codes. For instance, if s_1 corresponds to the Huffman code 00, s_2 corresponds to 1010, and s_3 corresponds to 110110, the compressed data for a sequence s_1 s_2 s_3 would be 010110010110.

Algorithm: Huffman Coding Compression

Inputs: S: Set of symbols in the input data; F: Frequency table containing the frequencies of symbols in S

Output: C: Huffman code dictionary mapping symbols to their Huffman codes; D: Compressed data obtained by replacing symbols in the input data with their Huffman codes

1) Initialize a priority queue Q to store nodes representing symbols and their frequencies.
   a) Insert each symbol s_i along with its frequency $f(s_i)$ into the priority queue Q.
2) While there is more than one node in the priority queue Q:
   a) Extract the two nodes with the lowest frequencies from Q.
      i) Create a new internal node with these two nodes as children, and assign the sum of their frequencies as the frequency of the new node.
      ii) Insert the new node back into Q.

b) The last remaining node in Q is the root of the Huffman tree.

3) Traverse the Huffman tree to assign Huffman codes to each symbol:
   a) Start from the root node and traverse down the tree.
      i) Assign '0' to edges leading to left child nodes and '1' to edges leading to right child nodes.
      ii) As traverse from the root to each leaf node, record the sequence of '0's and '1's, forming the Huffman code for each symbol.
   b) Store the Huffman codes in a dictionary C.
4) For each symbol s_i in the input data:
   a) Replace s_i with its corresponding Huffman code $C(s_i)$.
   b) Concatenate the Huffman codes for all symbols to obtain the compressed data D.

where,

S: Set of symbols in the input data

F: Frequency table containing the frequencies of symbols in S

Q: Priority queue storing nodes representing symbols and their frequencies

C: Huffman code dictionary mapping symbols to their Huffman codes

D: Compressed data obtained by replacing symbols in the input data with their Huffman codes

s_i: Symbol in the input data

$f(s_i)$: Frequency of symbol s_i in the input data

6. ATTACK CLASSIFICATION USING RNN

Algorithm: Attack Classification using RNN

Inputs: X: Input data consisting of network packet features; Y: Corresponding labels indicating the attack type for each input data sample; Num_ClassesNum_Classes: Number of attack classes; Num_FeaturesNum_Features: Number of features describing each packet

Output: Y': Predicted attack type for each input data sample

2) Normalize the input features X to ensure they have a mean of 0 and a standard deviation of 1.
   a) Convert the labels Y into one-hot encoded vectors, representing the attack classes.
3) Initialize an RNN model with appropriate parameters
   a) Configure the model architecture, specifying the input shape and output dimensionality.
   b) Compile the model with an appropriate loss function (categorical cross-entropy) and optimizer (Adam).
4) Split the data into training and validation sets.
   a) Train the RNN model using the training data:
      i) Feed the input data X_t into the model along with their corresponding labels Y_t.
      ii) Update the model parameters using backpropagation and gradient descent optimization.
      iii) Monitor the model's performance on the validation set to prevent overfitting.
5) Evaluate the trained model's performance on the test set:
   a) Feed the input data X_t into the trained model to obtain predictions Y'_t.
   b) Compare the predicted labels Y'_t with the ground truth labels Y_t
6) Deploy the trained model for real-time attack classification:
   a) Feed new input data into the model to obtain predictions Y'.

where,

X: Input data consisting of network packet features

Y: Corresponding labels indicating the attack type for each input data sample

Y': Predicted attack type for each input data sample

Num_ClassesNum_Classes: Number of attack classes

Num_FeaturesNum_Features: Number of features describing each packet

7. EXPERIMENTAL VALIDATION

For evaluating the proposed attack classification using Recurrent Neural Networks (RNNs), we conducted experiments using a simulation tool implemented in Python, leveraging libraries such as TensorFlow or PyTorch for building and training the RNN model. The dataset used for experiments consists of network packet features extracted from simulated network traffic, including features such as packet size, source and destination IP addresses, protocol type, and timestamp. The dataset is divided into training, validation, and test sets with a ratio of 70:15:15. We utilized a desktop computer with an Intel Core i7 processor, 16GB of RAM, and a Nvidia GeForce GTX GPU for model training and evaluation.

In our experiments, we compared the performance of the RNN-based attack classification method with existing methods such as Secure Socket Layer (SSL), Role-Based Access Control (RBAC), Datagram Transport Layer Security (DTLS), and Access Control Lists (ACL). We evaluated the models based on metrics such as accuracy, precision, recall, and F1-score.

Table.1. Setup

Parameter	Value
Simulation Tool	Python with TensorFlow
Dataset	Synthetic network traffic data
Dataset Split	70:15:15
Network Packet Features	Packet size, source IP, destination IP, protocol type, timestamp
Model Architecture	LSTM (Long Short-Term Memory)
Loss Function	Categorical cross-entropy
Optimizer	Adam
Learning Rate	0.001
Batch Size	64
Number of Epochs	50
Hardware	Intel Core i7 and 16GB RAM

Table.2. Detection Accuracy

Number of Nodes	SSL	RBAC	DTLS	ACL	Huffman Watermarking
50	0.85	0.78	0.82	0.79	0.92
100	0.88	0.82	0.85	0.81	0.94
150	0.90	0.85	0.87	0.83	0.95
200	0.91	0.86	0.88	0.85	0.96
250	0.92	0.88	0.89	0.87	0.97
300	0.93	0.89	0.90	0.88	0.97
350	0.93	0.90	0.91	0.89	0.98
400	0.94	0.91	0.92	0.90	0.98
450	0.94	0.92	0.92	0.91	0.99
500	0.95	0.92	0.93	0.92	0.99

Table.3. False Positive Rate

Number of Nodes	SSL	RBAC	DTLS	ACL	Huffman Watermarking
50	0.03	0.02	0.02	0.03	0.01
100	0.02	0.03	0.02	0.04	0.01
150	0.02	0.03	0.03	0.05	0.01
200	0.02	0.04	0.03	0.06	0.01
250	0.03	0.04	0.04	0.07	0.01
300	0.03	0.05	0.04	0.08	0.01
350	0.04	0.05	0.05	0.09	0.01
400	0.04	0.06	0.05	0.10	0.01
450	0.05	0.06	0.06	0.11	0.01
500	0.05	0.07	0.06	0.12	0.01

Table.4. Response Time (ms)

Number of Nodes	SSL	RBAC	DTLS	ACL	Huffman Watermarking
50	12.5	13.2	11.8	14.1	10.2
100	13.8	14.3	12.6	15.5	10.5
150	14.7	15.1	13.2	16.2	10.8
200	15.5	16.2	13.9	17.0	11.1
250	16.3	17.4	14.6	18.2	11.4
300	17.1	18.1	15.3	19.0	11.7
350	18.2	19.3	16.1	20.2	12.0
400	19.5	20.5	17.0	21.5	12.3
450	20.3	21.7	17.8	22.8	12.6
500	21.1	22.5	18.5	23.5	12.9

Table.5. Network Throughput (MBPS)

Number of Nodes	SSL	RBAC	DTLS	ACL	Huffman Watermarking
50	120	115	118	110	125
100	118	112	116	108	128
150	115	110	114	106	130
200	112	108	112	104	132
250	110	106	110	102	134
300	108	104	108	100	136
350	106	102	106	98	138
400	104	100	104	96	140
450	102	98	102	94	142
500	100	96	100	92	144

Table.5. Resource Utilization

Node	SSL	RBA	DTL	ACL	HW	SSL	RBA	DTL	ACL	HW
	CPU (%)					Memory (%)				
50	45	42	48	40	35	55	52	58	50	45
100	47	44	50	42	36	57	54	60	52	46
150	49	46	52	44	37	59	56	62	54	47
200	51	48	54	46	38	61	58	64	56	48
250	53	50	56	48	39	63	60	66	58	49
300	55	52	58	50	40	65	62	68	60	50
350	57	54	60	52	41	67	64	70	62	51
400	59	56	62	54	42	69	66	72	64	52
450	61	58	64	56	43	71	68	74	66	53
500	63	60	66	58	44	73	70	76	68	54

The detection accuracy of the proposed Huffman Watermarking method consistently outperformed existing methods across varying numbers of nodes. As the number of nodes increased, the accuracy of all methods generally improved, which can be attributed to the larger amount of data available for training and testing. However, the Huffman Watermarking method consistently exhibited the highest accuracy, reaching up to 99% accuracy for 500 nodes. This superior performance can be attributed to the robustness of the Huffman coding technique in accurately detecting and classifying attacks in network traffic. In contrast, existing methods such as SSL, RBAC, DTLS, and ACL showed lower accuracy rates, indicating their limitations in effectively identifying and mitigating network threats.

In terms of false positive rate, the proposed Huffman Watermarking method also demonstrated significant advantages over existing methods. The false positive rate remained consistently low for the Huffman Watermarking method across different numbers of nodes, indicating its ability to minimize the occurrence of false alarms or incorrect detections. In contrast, existing methods exhibited higher false positive rates, particularly as the number of nodes increased. This suggests that SSL, RBAC, DTLS, and ACL may be prone to misidentifying benign network activity as malicious attacks, leading to unnecessary alerts and resource consumption. The low false positive rate of the Huffman Watermarking method is particularly advantageous in real-world network security scenarios, where minimizing false alarms is critical for efficient threat detection and response.

Resource utilization, including CPU and memory usage, is another important factor to consider in network security solutions.

The results show that the proposed Huffman Watermarking method achieved lower resource utilization compared to existing methods. Across different numbers of nodes, the CPU and memory utilization percentages for the Huffman Watermarking method were consistently lower than those of SSL, RBAC, DTLS, and ACL. This indicates that the Huffman Watermarking method is more efficient in terms of resource consumption, making it well-suited for deployment in resource-constrained environments or high-throughput networks. Lower resource utilization not only reduces operational costs but also enhances the scalability and performance of the network security infrastructure.

The superior performance of the proposed Huffman Watermarking method in terms of detection accuracy, false positive rate, and resource utilization has significant practical implications for network security practitioners and organizations. By adopting the Huffman Watermarking method, organizations can enhance their ability to detect and classify network attacks accurately while minimizing false alarms and conserving computational resources. This can lead to improved threat response times, reduced operational overhead, and enhanced overall security posture. Additionally, the efficiency and scalability of the Huffman Watermarking method make it suitable for deployment in diverse network environments, ranging from small-scale enterprise networks to large-scale data centers and cloud infrastructures.

8. CONCLUSION

The results of the experiments highlight the effectiveness and efficiency of the proposed Huffman Watermarking method for network security applications. By outperforming existing methods in terms of detection accuracy, false positive rate, and resource utilization, the Huffman Watermarking method offers a promising approach for mitigating network threats and enhancing overall security resilience. Future research directions may involve further optimization of the Huffman coding technique, exploration of hybrid approaches combining watermarking with other machine learning methods, and real-world validation of the proposed method in diverse network environments.

The experiments conducted to compare existing network security methods (SSL, RBAC, DTLS, ACL) with the proposed Huffman Watermarking method across varying numbers of nodes have yielded insightful results. The findings underscore the effectiveness and efficiency of the Huffman Watermarking method in enhancing network security and mitigating cyber threats.

- The Huffman Watermarking method consistently exhibited higher detection accuracy compared to existing methods, reaching up to 99% accuracy for 500 nodes. This superior performance highlights the robustness of Huffman coding in accurately classifying network attacks.
- The Huffman Watermarking method showed lower CPU and memory utilization percentages compared to existing methods. This efficiency in resource consumption enhances scalability and reduces operational costs, making the proposed method well-suited for deployment in various network environments.

REFERENCES

[1] Haibo Hu and Jianliang Xu, "2PASS: Bandwidth-Optimized Location Cloaking for Anonymous Location-Based Services", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, No. 10, pp. 1458-1472, 2010.

[2] Ping Yi, Zhoulin Dai, Shiyong Zhang and Yiping Zhong, "A New Routing Attack in Mobile Ad Hoc Networks", *International Journal of Information Technology*, Vol. 11, No. 2, pp. 83-94, 2005.

[3] Zhiqiang Gao and Zhiqiang, "Differentiating Malicious DDoS Attack Traffic from Normal TCP Flows by Proactive Tests", *IEEE Communications Letters*, Vol. 10, No. 11, pp. 793-795, 2006.

[4] M. Zhou, L. Han, H. Lu and C. Fu, "Intrusion Detection System for IoT Heterogeneous Perceptual Network", *Mobile Networks and Applications*, Vol. 33, No. 1, pp. 1-14, 2020.

[5] L. Xiao, X. Wan, X. Lu and Y. Zhang, "IoT Security Techniques based on Machine Learning: How do IoT Devices use AI to Enhance Security?", *IEEE Signal Processing Magazine*, Vol. 35, No. 5, pp. 41-49, 2018.

[6] B. Gobinathan and V.P. Sundramurthy, "A Novel Method to Solve Real Time Security Issues in Software Industry using Advanced Cryptographic Techniques", *Scientific Programming*, Vol. 2021, pp. 1-9, 2021.

[7] Qi Liao, David A. Cieslak, Aaron D. Striegel and Nitesh V. Chawla, "Using Selective, Short-Term Memory to Improve Resilience against DDoS Exhaustion Attack", *Security and Communication Networks*, Vol. 1, pp. 287-299, 2008.

[8] J. Singh, R. Vikram and S. Sakthivel, "Energy-Efficient Clustering and Routing Algorithm using Hybrid Fuzzy with Grey Wolf Optimization in Wireless Sensor Networks", *Security and Communication Networks*, Vol. 2022, pp. 1-13, 2022.

[9] V. Saravanan and A. Sumathi, "Handoff Mobiles with Low Latency in Heterogeneous Networks for Seamless Mobility: A Survey and Future Directions", *European Journal of Scientific Research*, Vol. 81, No. 3, pp. 417-424, 2012.

[10] J. Lloyd and R. Anane, "Implementation of A System for Cohesive and Secure Community Management", *Proceedings of International Conference on Computer Supported Cooperative Work in Design*, pp. 133-138, 2021.

[11] S.P. Yadav, F. Al Turjman and S.A. Kumar, "Quantum-Safe Cryptography Algorithms and Approaches: Impacts of Quantum Computing on Cybersecurity", De Gruyter, 2023.

[12] R.N. Shanmugasundaram, "Enhancements of Resource Management for Device to Device (D2D) Communication: A Review", *Proceedings of International Conference on IoT in Social, Mobile, Analytics and Cloud*, pp. 51-55, 2019.