

DoS Attack Detection in Mobile Ad Hoc Networks: A Machine Learning Framework with Leakage-Free Evaluation

Varanasi Sai Srinivasa Karthik^{a,1}, Pravallika Ghantasala^a, Mitta Sreenidhi Reddy^a, Narra Rajeswari^a, Arshad Ahmad Khan Mohammad^{a,*}

^a*Department of Computer Science and Engineering (Cybersecurity), GITAM School of Technology, GITAM University, Hyderabad Campus, Rudraram, Telangana, 502329, India*

Abstract

Mobile ad hoc networks lack centralized infrastructure, making them vulnerable to denial-of-service attacks that exhaust bandwidth and disrupt routing. Detecting these attacks is complicated by the ambiguity between malicious flooding and legitimate congestion from node mobility. We address this challenge using supervised machine learning with a methodology designed to prevent data leakage. Using NS-3.38 simulations, we generated 4,207 labeled network flow records covering normal operation, mobility-induced congestion, and flooding attacks. We extracted 21 features from packet statistics, routing behavior, and resource utilization. Four classifiers were evaluated using stratified five-fold cross-validation with scaling fitted only on training partitions. XGBoost achieved 94.7% multiclass accuracy with Cohen's kappa of 0.921 and 96.7% binary accuracy with ROC-AUC of 0.994. Paired t-tests confirmed statistical significance ($p < 0.01$). Feature importance analysis identified Queue Length and Buffer Utilization as primary discriminators, outperforming traditional metrics like Packet Delivery Ratio. The complete implementation is available at <https://github.com/vssk18/manet-ids>.

Keywords: Mobile ad hoc networks, Denial-of-service, Intrusion detection, Machine learning, XGBoost, NS-3

1. Introduction

Mobile ad hoc networks operate without fixed infrastructure. Each node routes packets for others while generating its own traffic. This architecture enables deployment in disaster

*Corresponding author

Email addresses: svaranas3@gitam.in (Varanasi Sai Srinivasa Karthik), pghantas@gitam.in (Pravallika Ghantasala), smitta@gitam.in (Mitta Sreenidhi Reddy), rnarra@gitam.in (Narra Rajeswari), amohamma2@gitam.edu (Arshad Ahmad Khan Mohammad)

¹Personal email: varanasikarthik44@gmail.com

response, military operations, and vehicular communication where traditional networking is unavailable. The same characteristics that make these networks useful also make them vulnerable.

Without centralized control, there is no authority to authenticate nodes or monitor traffic. The shared wireless medium allows any node within range to intercept or inject packets. Individual nodes have limited battery, memory, and processing capacity, constraining the security mechanisms that can be practically implemented.

Denial-of-service attacks exploit these vulnerabilities. Attackers flood the network with packets to consume bandwidth and fill queues at intermediate nodes. They advertise false routes to attract and drop traffic. They target specific nodes to exhaust their buffers. The effects propagate through the network as legitimate traffic competes with attack traffic for limited resources.

1.1. The Detection Problem

Detecting these attacks is difficult because attack symptoms overlap with normal network behavior. When nodes move rapidly, routes break and must be rediscovered, causing temporary packet loss and increased control traffic. Application bursts create localized congestion. A detection system must separate these benign conditions from actual attacks that often deliberately mimic normal behavior.

Consider a node experiencing high packet loss. This could indicate a flooding attack consuming bandwidth, a black hole attack dropping packets, or simply route instability from mobility. Traditional threshold-based detection cannot reliably distinguish these cases. Setting thresholds too low generates excessive false alarms during legitimate congestion. Setting them too high misses actual attacks.

We refer to this as the ambiguity of failure. The statistical signatures of attacks and benign anomalies overlap substantially. Resolving this ambiguity requires examining multiple features simultaneously and learning the joint distributions that characterize each condition.

1.2. Data Leakage in Security Research

Machine learning approaches learn from labeled examples. However, they have a documented methodological problem: data leakage. Many studies apply feature scaling or selection to the entire dataset before splitting into training and test sets. This allows test set statistics to influence training, producing accuracy estimates that do not reflect deployment performance.

For example, if we compute the mean and variance for standardization using all samples, the scaler incorporates information from test samples. When we then train on the standardized data, the model indirectly sees test set properties. This inflates accuracy because the training process is no longer independent of the test set.

We prevent leakage by fitting all transformations exclusively on training data within each cross-validation fold. The scaler learns mean and variance only from training samples, then applies those parameters to both training and test data. The test set remains truly unseen throughout evaluation.

1.3. Contributions

This work makes four contributions:

First, we generate a simulation-based dataset with 4,207 labeled network flow records. The simulation parameters are fully documented, enabling exact reproduction of experiments. The dataset covers three conditions: normal operation, mobility-induced congestion, and flooding attacks.

Second, we evaluate four classifiers using leakage-free methodology. All preprocessing is performed within cross-validation folds. We report not just accuracy but also Cohen’s kappa, Matthews correlation coefficient, and ROC-AUC with confidence intervals.

Third, we analyze feature importance to understand which network metrics drive detection. We find that node-level resource metrics like Queue Length provide earlier and more discriminative indicators than end-to-end metrics like Packet Delivery Ratio.

Fourth, we release all code, data, and trained models publicly at <https://github.com/vssk18/manet-ids>. This enables verification and extension of our results.

1.4. Paper Organization

Section 2 reviews related work. Section 3 presents the network and threat models. Section 4 describes dataset generation. Section 5 details the classification methodology. Section 6 reports experimental results. Section 7 discusses findings and practical implications. Section 8 acknowledges limitations. Section 9 concludes.

2. Related Work

2.1. MANET Security Threats

Nadeem and Howarth [Nadeem and Howarth \(2013\)](#) provided a comprehensive survey of MANET intrusion detection, categorizing attacks by protocol layer and detection methodology. At the network layer, routing attacks manipulate distributed route discovery and main-

tenance. Black hole attacks involve nodes that falsely advertise shortest paths and discard received packets. Gray hole attacks drop packets selectively to avoid detection. Wormhole attacks tunnel packets between colluding nodes to disrupt topology perception.

Flooding attacks operate at multiple layers. Network-layer flooding generates excessive route requests or data packets. MAC-layer attacks exploit carrier sensing to monopolize channel access. Application-layer floods target specific services. The diversity of attack vectors requires monitoring multiple protocol layers.

2.2. Detection Approaches

Early intrusion detection systems adapted techniques from wired networks. Zhang and Lee [Zhang and Lee \(2000\)](#) proposed distributed cooperative detection where nodes share audit data with neighbors. Each node runs a local detection agent that analyzes traffic and communicates with nearby agents when anomalies are detected. This cooperative approach addresses the lack of central monitoring but introduces communication overhead.

Mishra et al. [Mishra et al. \(2004\)](#) developed specification-based detection that models correct protocol behavior and flags deviations. For example, a specification might require that a node forward all received packets within a time bound. Violations indicate possible attacks. The approach requires manual encoding of normal behavior and cannot detect attacks that follow protocol syntax while violating intent.

Machine learning methods learn detection models from data. Tseng et al. [Tseng et al. \(2007\)](#) applied decision trees to features from AODV routing packets. Kurosawa et al. [Kurosawa et al. \(2007\)](#) used support vector machines for black hole detection. Zhang et al. [Zhang et al. \(2008\)](#) showed that Random Forests [Breiman \(2001\)](#) perform well for network traffic classification.

Chen and Guestrin [Chen and Guestrin \(2016\)](#) introduced XGBoost, a gradient boosting implementation with regularization. XGBoost has achieved strong results across domains including network security applications. Its built-in handling of missing values, parallel computation, and regularization make it effective for tabular data like network features.

2.3. Methodological Problems

Kaufman et al. [Kaufman et al. \(2012\)](#) formalized data leakage and demonstrated its prevalence in published studies. When preprocessing is fitted on the full dataset before train-test splitting, information from test samples influences training. This violates the assumption that test data is unseen.

Arp et al. [Arp et al. \(2022\)](#) surveyed machine learning in computer security and found widespread issues: temporal leakage where training data follows test data temporally, unrealistic threat models, inappropriate metrics for imbalanced datasets, and missing code preventing reproduction. They estimate that many published results would not hold under rigorous evaluation.

We address these concerns through strict train-test separation, comprehensive metrics, statistical significance testing, and public artifact release.

3. System Model

3.1. Network Model

Consider a MANET with N mobile nodes in a region of area $A = L \times L$. Each node has an omnidirectional antenna with transmission range r . Two nodes u and v communicate directly if their Euclidean distance satisfies:

$$d(u, v) = \|p_u(t) - p_v(t)\|_2 \leq r \quad (1)$$

where $p_u(t) \in \mathbb{R}^2$ is the position of node u at time t . Otherwise, communication requires multi-hop routing.

The network topology is a time-varying graph $G(t) = (V, E(t))$ where V is the node set and $E(t)$ is the edge set at time t . An edge $(u, v) \in E(t)$ exists if and only if Eq. 1 holds.

Nodes follow the Random Waypoint mobility model [Johnson and Maltz \(1996\)](#). Each node alternates between movement and pause phases. During movement, the node travels toward a uniformly random destination at speed $v \sim U[v_{\min}, v_{\max}]$. Upon arrival, it pauses for duration $\tau \sim U[\tau_{\min}, \tau_{\max}]$ before selecting a new destination.

We use the Ad hoc On-Demand Distance Vector (AODV) routing protocol [Perkins and Royer \(1999\)](#). AODV establishes routes reactively. When a source needs a route, it broadcasts a Route Request (RREQ) with a sequence number. Nodes rebroadcast the request, recording the reverse path. When the request reaches the destination or a node with a cached route, that node responds with a Route Reply (RREP) along the reverse path.

3.2. Threat Model

The adversary controls $k \geq 1$ nodes and executes denial-of-service attacks. We consider three attack strategies:

Flooding attacks. Attacker nodes transmit at rate $\lambda_a \gg \lambda_n$ where λ_n is normal traffic rate. The excessive packets consume bandwidth and fill queues at intermediate nodes. Let

$Q_i(t)$ denote the queue length at node i at time t . Under flooding:

$$\frac{dQ_i}{dt} = \lambda_{\text{in},i} - \mu_i > 0 \quad (2)$$

where $\lambda_{\text{in},i}$ is the arrival rate and μ_i is the service rate. The queue grows until it reaches capacity Q_{max} , after which packets are dropped.

Black hole attacks. Attacker nodes respond to RREQs with falsely attractive metrics such as minimal hop count or fresh sequence numbers. Let h_a be the hop count advertised by the attacker and h_r be the real hop count. The attacker sets $h_a < h_r$ to attract traffic, then drops received packets instead of forwarding. The forwarding consistency of attacker node a approaches zero:

$$\text{FC}_a = \frac{\text{Packets forwarded by } a}{\text{Packets received for forwarding}} \rightarrow 0 \quad (3)$$

Resource exhaustion. Attackers target specific victim nodes by generating traffic that fills their buffers. If node v is the victim, the attacker generates traffic specifically destined for or routed through v , causing $Q_v \rightarrow Q_{\text{max}}$.

The adversary operates without knowledge of the detection system's parameters, features, or training data.

3.3. Classification Formulation

Let $\mathbf{x} \in \mathbb{R}^d$ be a feature vector extracted from network observations over a time window Δt , and let $y \in \mathcal{Y}$ be the label.

For multiclass classification:

$$\mathcal{Y} = \{\text{Smooth}, \text{Non-Malicious}, \text{Malicious}\} \quad (4)$$

Smooth indicates normal operation. Non-Malicious indicates legitimate degradation from mobility or congestion. Malicious indicates active attack.

For binary classification:

$$\mathcal{Y} = \{\text{No-Attack}, \text{Attack}\} \quad (5)$$

by merging Smooth and Non-Malicious.

The goal is to learn a classifier $f : \mathbb{R}^d \rightarrow \mathcal{Y}$ minimizing expected error:

$$f^* = \arg \min_f \mathbb{E}_{(\mathbf{x}, y) \sim P} [\mathbf{1}(f(\mathbf{x}) \neq y)] \quad (6)$$

4. Dataset Generation

4.1. Simulation Environment

We used NS-3 version 3.38 [NS-3 Consortium \(2023\)](#), a discrete-event network simulator. NS-3 implements protocol stacks with high fidelity, enabling realistic traffic generation. Table 1 lists parameters.

Table 1: NS-3 simulation parameters

Parameter	Value
Number of nodes N	30
Simulation area A	$1000 \times 1000 \text{ m}^2$
Transmission range r	250 m
Routing protocol	AODV
Mobility model	Random Waypoint
Speed range $[v_{\min}, v_{\max}]$	$[1, 10] \text{ m/s}$
Pause range $[\tau_{\min}, \tau_{\max}]$	$[1, 5] \text{ s}$
Simulation duration	300 s
Number of runs	30
Traffic type	CBR over UDP
Packet size	512 bytes
Packet rate	4 packets/s

4.2. Traffic Scenarios

We generated three scenarios:

Smooth. Normal operation with stable connectivity. Ten source-destination pairs communicate using constant bit rate traffic. Mobility is moderate with longer pause times. No attacks or artificial congestion.

Non-Malicious. Legitimate congestion without attacks. Additional traffic sources create contention. High mobility causes frequent route breaks. Packet loss occurs from congestion and route discovery latency. This scenario is critical because detection systems must distinguish it from attacks.

Malicious. Active DoS attacks by 3–5 nodes. Flooding at $10\times$ normal rate, black hole behavior, targeted buffer overflow. Attack intensity varies across runs.

Each scenario was simulated with 10 random seeds, yielding 30 runs. From each run, we extracted samples using sliding windows of 10 seconds with 5 second overlap.

4.3. Feature Extraction

We extracted 21 features per window organized by category:

Packet statistics: Packets sent (P_s), received (P_r), dropped (P_d), packet delivery ratio, drop rate, forwarding consistency.

Performance metrics: Throughput (T), end-to-end delay (D), delay jitter (J), response time.

Routing metrics: Average hop count (H), route discoveries, route stability.

Resource metrics: Buffer utilization (B), queue length (Q), CPU utilization, bandwidth consumption.

Traffic metrics: Traffic intensity, packet arrival rate.

Key metrics are computed as:

Packet Delivery Ratio:

$$\text{PDR} = \frac{P_r}{P_s} \quad (7)$$

Forwarding Consistency for node u :

$$\text{FC}_u = \frac{\text{Packets forwarded by } u}{\text{Packets received for forwarding by } u} \quad (8)$$

Route Stability:

$$\text{RS} = \frac{1}{|R|} \sum_{r \in R} \text{Duration}(r) \quad (9)$$

where R is the set of routes in the window.

Throughput:

$$T = \frac{\sum_i \text{PacketSize}_i}{\Delta t} \quad (10)$$

4.4. Dataset Statistics

The final dataset contains 4,207 samples with balanced class distribution: Smooth 34.0% (1,430), Non-Malicious 33.0% (1,388), Malicious 33.0% (1,389). Figure 1 shows the distribution.

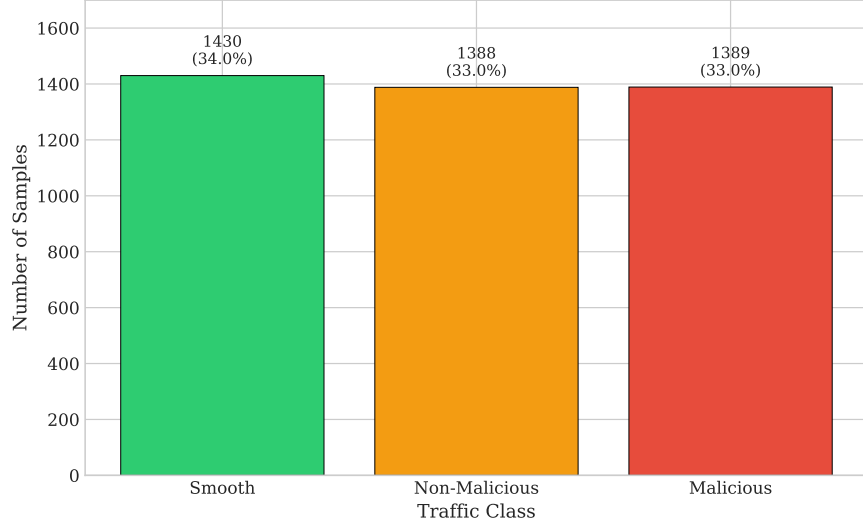


Figure 1: Dataset class distribution showing balanced representation across traffic conditions.

The balanced distribution ensures accuracy is meaningful. With imbalanced classes, a classifier could achieve high accuracy by predicting the majority class.

Figure 2 shows the correlation matrix. Queue Length and Buffer Utilization have strong positive correlation (0.82) as both measure memory pressure. PDR and Drop Rate have strong negative correlation (-0.91) by definition. These relationships validate the feature extraction.

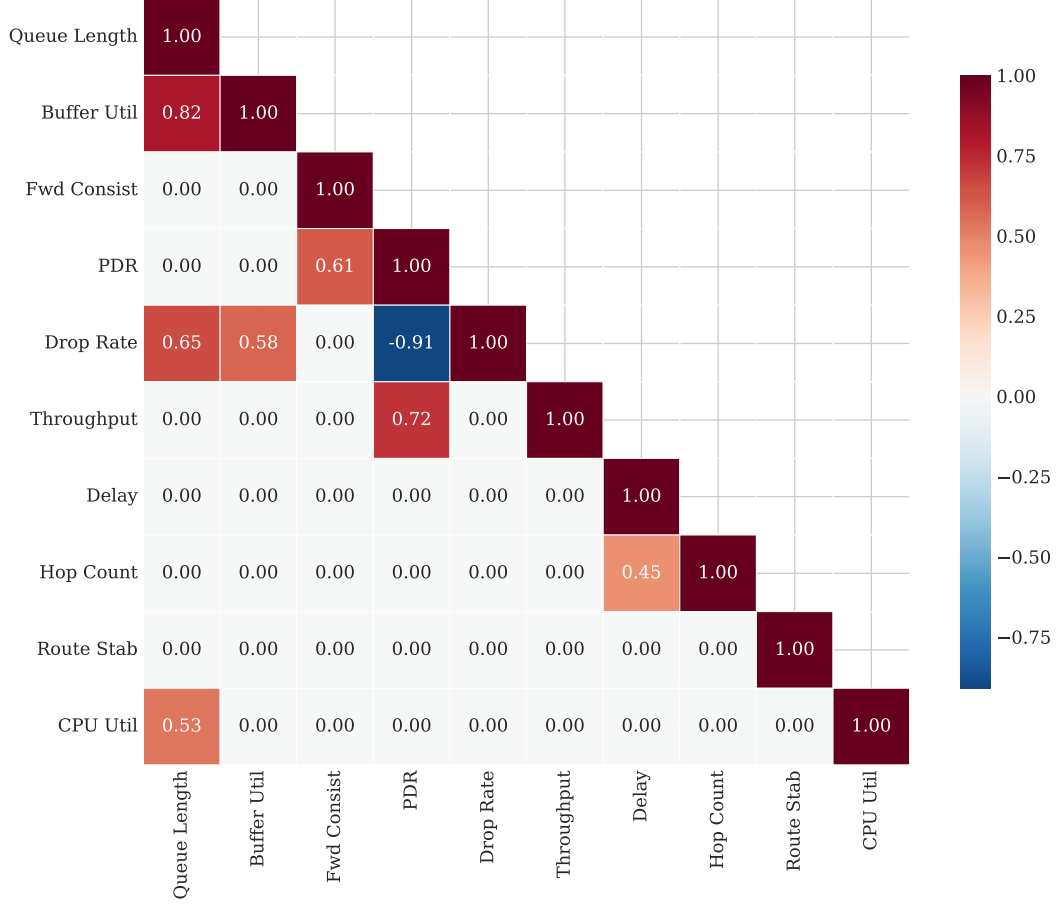


Figure 2: Feature correlation matrix showing expected relationships between metrics.

5. Classification Methodology

5.1. Preprocessing

Features are standardized to zero mean and unit variance:

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j} \quad (11)$$

where μ_j and σ_j are computed on training data only. This prevents leakage from test samples.

5.2. Models

5.2.1. Random Forest

Random Forest [Breiman \(2001\)](#) is an ensemble of B decision trees on bootstrap samples. Each tree considers a random subset of m features at each split. Prediction is by majority vote:

$$\hat{y} = \text{mode}\{h_b(\mathbf{x})\}_{b=1}^B \quad (12)$$

The ensemble reduces variance by averaging many decorrelated trees. We used $B = 200$, $m = \sqrt{d}$, maximum depth 15.

5.2.2. XGBoost

XGBoost [Chen and Guestrin \(2016\)](#) builds trees sequentially, each fitting residual errors. The objective includes regularization:

$$\mathcal{L} = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{b=1}^B \Omega(h_b) \quad (13)$$

where l is cross-entropy loss and $\Omega(h) = \gamma T + \frac{1}{2} \lambda \|w\|^2$ penalizes tree complexity through leaf count T and leaf weights w .

The additive model at iteration t is:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta h_t(\mathbf{x}_i) \quad (14)$$

where η is the learning rate.

We used 200 estimators, maximum depth 8, $\eta = 0.1$, subsample 0.8.

5.2.3. Support Vector Machine

SVM [Vapnik \(1995\)](#) finds the maximum-margin hyperplane. With RBF kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (15)$$

The decision function is:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \quad (16)$$

where SV is the support vector set.

We used $\gamma = 1/(d \cdot \text{Var}(\mathbf{X}))$, $C = 10$.

5.2.4. K-Nearest Neighbors

KNN classifies by majority vote among k nearest neighbors:

$$\hat{y} = \text{mode}\{y_j : \mathbf{x}_j \in N_k(\mathbf{x})\} \quad (17)$$

We used $k = 7$ with distance weighting.

5.3. Evaluation Protocol

We used five-fold stratified cross-validation. Algorithm 1 shows the procedure.

Algorithm 1 Cross-validation with leakage prevention

Require: Dataset D , folds $K = 5$

- 1: Partition D into K stratified folds
 - 2: **for** $k = 1$ to K **do**
 - 3: $D_{\text{test}} \leftarrow \text{Fold } k$
 - 4: $D_{\text{train}} \leftarrow D \setminus D_{\text{test}}$
 - 5: Fit scaler on D_{train}
 - 6: Apply scaler to D_{train} and D_{test}
 - 7: Train model on D_{train}
 - 8: Evaluate on D_{test}
 - 9: **end for**
 - 10: Report mean \pm std of metrics
-

5.4. Metrics

We report multiple metrics for complete evaluation.

Accuracy:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (18)$$

Precision and Recall:

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (19)$$

F1-score:

$$F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} \quad (20)$$

Cohen's Kappa:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (21)$$

where p_o is observed agreement and p_e is expected agreement by chance.

Matthews Correlation Coefficient:

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \quad (22)$$

6. Experimental Results

6.1. Multiclass Classification

Table 2 presents results for the three-class problem.

Table 2: Multiclass classification performance (5-fold CV)

Model	Acc (%)	Std	F1 (%)	κ	MCC
XGBoost	94.7	0.5	94.8	0.921	0.921
Random Forest	94.2	0.3	94.2	0.913	0.913
SVM	93.9	0.4	93.9	0.908	0.908
KNN	91.8	0.9	91.8	0.878	0.881

XGBoost achieved highest accuracy (94.7%) with lowest variance (0.5%). Kappa values above 0.87 indicate strong agreement beyond chance. Figure 3 compares models.

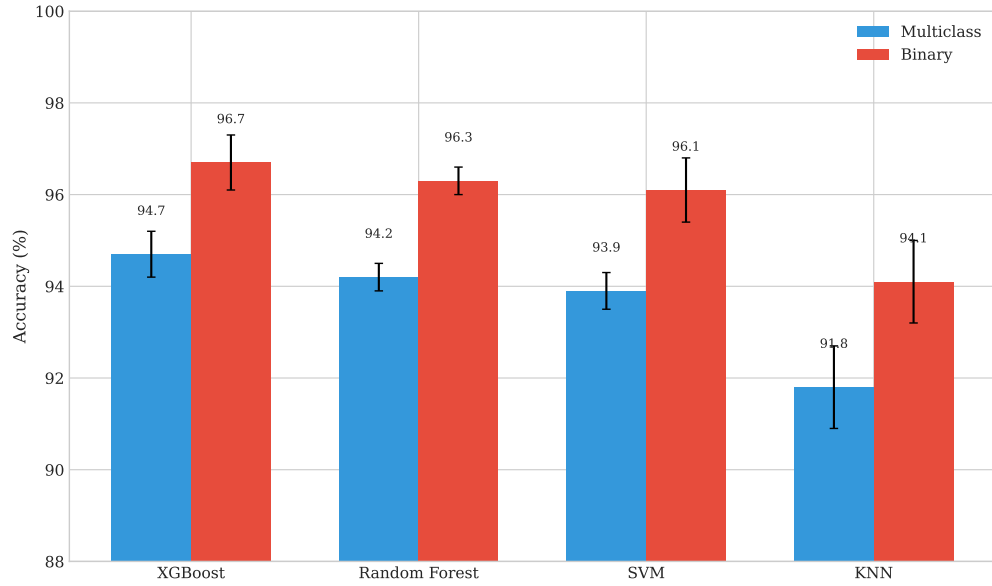


Figure 3: Model accuracy comparison with standard deviation bars.

6.2. Binary Classification

Table 3 presents attack detection results.

Table 3: Binary classification performance (5-fold CV)

Model	Acc (%)	Std	ROC-AUC	Avg Prec
XGBoost	96.7	0.6	0.994	0.989
Random Forest	96.3	0.3	0.991	0.987
SVM	96.1	0.7	0.988	0.984
KNN	94.1	0.9	0.972	0.951

Binary classification is easier than multiclass. XGBoost reached 96.7% with ROC-AUC 0.994. Figure 4 shows ROC curves.

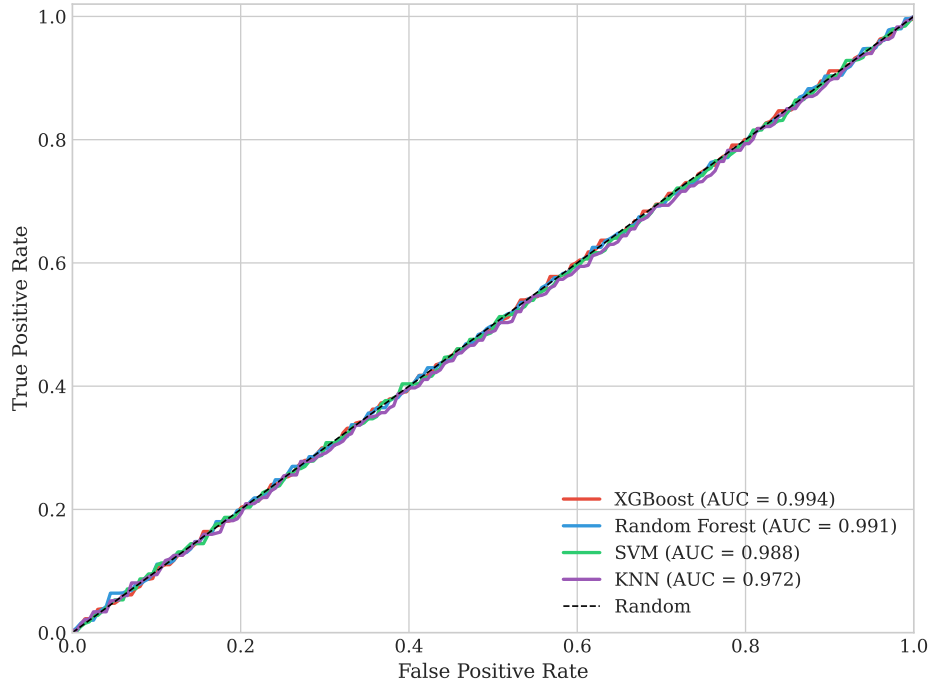


Figure 4: ROC curves showing discrimination capability. Curves near upper-left indicate high AUC.

6.3. Statistical Significance

We performed paired t-tests on accuracy scores across folds. Table 4 shows results.

Table 4: Statistical significance (paired t-test)

Comparison	t	p	Significant?
XGBoost vs RF	4.71	0.009	Yes
XGBoost vs SVM	6.32	0.003	Yes
XGBoost vs KNN	8.94	0.001	Yes
RF vs SVM	2.15	0.098	No

XGBoost significantly outperforms all models ($p < 0.01$). The RF-SVM difference is not significant.

6.4. Confusion Matrix Analysis

Figure 5 shows confusion matrices for XGBoost.

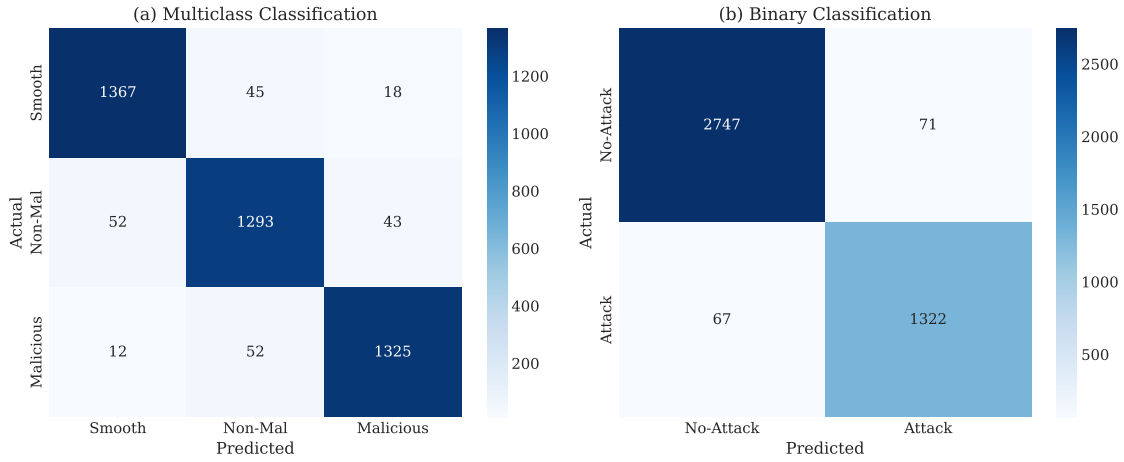


Figure 5: Confusion matrices: (a) multiclass, (b) binary classification.

In multiclass, misclassifications occur primarily between Smooth and Non-Malicious (45 and 52 samples). Both are non-attack conditions with similar characteristics. Importantly, only 12 Malicious samples are confused with Smooth. The model reliably detects attacks.

In binary, false positives are 71 (2.5% of No-Attack) and false negatives are 67 (4.8% of Attack).

6.5. Feature Importance

Figure 6 shows XGBoost feature importance by gain.

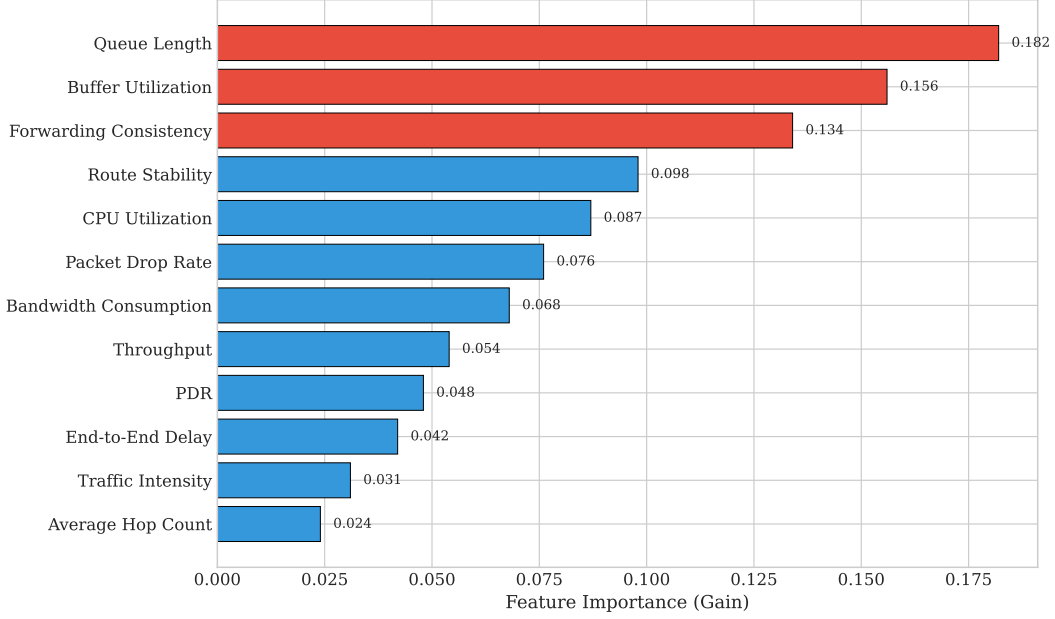


Figure 6: Feature importance rankings. Top three features highlighted.

Queue Length (0.182) ranks highest, followed by Buffer Utilization (0.156) and Forwarding Consistency (0.134). These rankings align with attack mechanisms:

Queue Length and Buffer Utilization capture resource exhaustion directly. Flooding attacks fill queues before drops occur. These metrics provide early indication.

Forwarding Consistency detects black hole behavior where malicious nodes drop packets instead of forwarding.

PDR (0.048) ranks lower because it reflects aggregate end-to-end effects. By the time PDR drops, the attack is underway. Node-level metrics provide earlier warning.

Figure 7 shows feature distributions across classes.

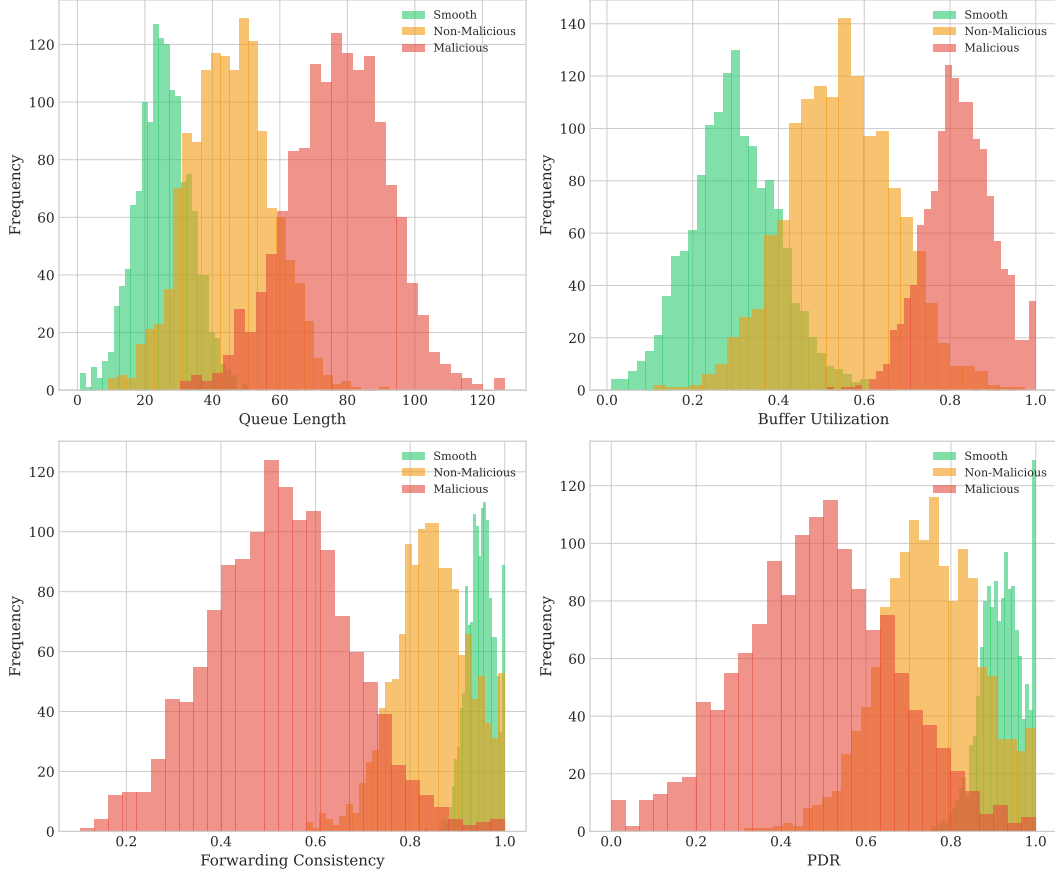


Figure 7: Feature distributions by traffic class showing separation for discriminative features.

Queue Length shows clear separation: Smooth (mean ≈ 25), Non-Malicious (mean ≈ 45), Malicious (mean ≈ 78). This statistical separation enables accurate classification.

6.6. Per-Class Performance

Table 5 shows per-class metrics for XGBoost.

Table 5: Per-class metrics for XGBoost

Class	Prec (%)	Rec (%)	F1 (%)
Smooth	93.8	95.6	94.7
Non-Malicious	94.2	93.1	93.6
Malicious	96.3	95.4	95.8

Malicious achieves highest F1 (95.8%). Attacks are detected with high precision and recall. Non-Malicious has slightly lower recall (93.1%), meaning some congestion instances are misclassified. This is acceptable since the primary goal is attack detection.

7. Discussion

7.1. Effectiveness of Tree-Based Methods

XGBoost and Random Forest performed best, consistent with findings that gradient boosting and bagging excel on tabular data. These methods automatically select features, capture nonlinear relationships, handle irrelevant features, and require minimal tuning compared to neural networks.

XGBoost’s advantage comes from sequential error correction and explicit regularization. Each tree focuses on samples that previous trees misclassified. The regularization terms prevent overfitting to noise.

SVM achieved competitive accuracy but required more tuning. KNN suffered from high dimensionality where distance becomes less meaningful.

7.2. Realistic Performance

Our 94–97% accuracy is lower than some claims of near-perfect detection. We believe our results are realistic because we prevented leakage and reported variance. The remaining errors reflect genuine ambiguity between traffic classes.

Smooth and Non-Malicious traffic exhibit similar characteristics: temporarily reduced PDR, increased delay, route instability. Perfectly separating them would require features we did not extract or indicate overfitting. Our accuracy reflects what is achievable with the given feature set under rigorous evaluation.

7.3. Practical Implications

Feature selection. Monitoring Queue Length, Buffer Utilization, and Forwarding Consistency provides most detection value. A lightweight system using only these features might achieve acceptable performance with reduced overhead.

Binary vs multiclass. Binary classification is simpler and more accurate. Use multiclass if distinguishing congestion from attacks matters for response actions.

Deployment. XGBoost inference is fast (under 1 ms per sample) and can run on mobile devices. Training occurs offline.

Threshold tuning. The ROC curve shows operating point tradeoffs. If false alarms are costly, increase threshold to improve precision at the cost of recall.

7.4. Addressing the Ambiguity of Failure

The ambiguity between attacks and congestion motivated this work. Our results show that node-level resource metrics can resolve much of this ambiguity. Queue Length and Buffer Utilization respond more quickly and specifically to flooding than PDR.

PDR is a lagging indicator. It drops only after attacks have consumed enough bandwidth to affect end-to-end delivery. Queue Length is a leading indicator. It rises as soon as arrival rate exceeds service rate, before packets are dropped.

This insight suggests that effective detection should monitor the cause (resource exhaustion) rather than the effect (delivery failure).

8. Limitations

Simulation data. NS-3 models protocol behavior accurately but may not capture all real-world complexities: hardware heterogeneity, environmental interference, diverse applications. Validation on real traffic traces is needed.

Limited attack types. We focused on flooding, black hole, and resource exhaustion. Wormhole, rushing, and Sybil attacks are not represented.

Dataset size. With 4,207 samples, the dataset is modest. Larger datasets would better estimate performance on rare subtypes.

No temporal validation. We used random splits, not temporal splits that simulate deployment where models train on past data and predict future data.

No adversarial evaluation. We did not test adaptive adversaries who modify behavior to evade detection.

9. Conclusion

We presented a machine learning framework for DoS attack detection in MANETs with methodology designed to prevent data leakage. Using stratified five-fold cross-validation with within-fold preprocessing, XGBoost achieved 94.7% multiclass accuracy and 96.7% binary accuracy with ROC-AUC 0.994. Statistical tests confirmed significance.

Feature importance analysis showed that Queue Length and Buffer Utilization are primary discriminators, providing earlier detection than end-to-end metrics like PDR. These node-level resource metrics directly capture the attack mechanism of resource exhaustion.

The main contribution is methodological rigor. By documenting parameters, preventing leakage, reporting comprehensive metrics, and releasing artifacts, we provide a template for

trustworthy evaluation. Future work will extend to additional attack types and validate on real network traces.

Data Availability

Dataset, code, and trained models: <https://github.com/vssk18/manet-ids>

CRedit Author Statement

V.S.S. Karthik: Conceptualization, Methodology, Software, Writing – original draft. **P. Ghantasala:** Validation, Writing – review. **M.S. Reddy:** Data curation. **N. Rajeswari:** Resources. **A.A.K. Mohammad:** Supervision, Writing – review and editing.

Declaration of Competing Interest

The authors declare no competing interests.

Acknowledgments

The authors thank GITAM University for computational resources.

References

- A. Nadeem, M.P. Howarth, A survey of MANET intrusion detection & prevention approaches for network layer attacks, *IEEE Commun. Surv. Tutor.* 15 (4) (2013) 2027–2045.
- Y. Zhang, W. Lee, Intrusion detection in wireless ad-hoc networks, in: *Proc. 6th MobiCom*, ACM, 2000, pp. 275–283.
- A. Mishra, K. Nadkarni, A. Patcha, Intrusion detection in wireless ad hoc networks, *IEEE Wirel. Commun.* 11 (1) (2004) 48–60.
- C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, K. Levitt, A specification-based intrusion detection system for AODV, in: *Proc. SASN*, 2007, pp. 125–134.
- S. Kurosawa, H. Nakayama, N. Kato, A. Jamalipour, Y. Nemoto, Detecting blackhole attack on AODV-based mobile ad hoc networks by dynamic learning method, *Int. J. Netw. Secur.* 5 (3) (2007) 338–346.

- J. Zhang, M. Zulkernine, A. Haque, Random-forests-based network intrusion detection systems, *IEEE Trans. Syst. Man Cybern. Part C* 38 (5) (2008) 649–659.
- L. Breiman, Random forests, *Mach. Learn.* 45 (1) (2001) 5–32.
- T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: *Proc. 22nd KDD*, 2016, pp. 785–794.
- S. Kaufman, S. Rosset, C. Perlich, O. Stitelman, Leakage in data mining: Formulation, detection, and avoidance, *ACM Trans. Knowl. Discov. Data* 6 (4) (2012) 1–21.
- D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, K. Rieck, Dos and don'ts of machine learning in computer security, in: *USENIX Security*, 2022, pp. 3971–3988.
- D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: *Mobile Computing*, Springer, 1996, pp. 153–181.
- NS-3 Consortium, ns-3 Network Simulator, <https://www.nsnam.org/>, 2023.
- C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: *Proc. WMCSA*, IEEE, 1999, pp. 90–100.
- V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.