Engineering Systems and Intelligent Technologies
ESIT
ISSN: 3071-253X/
Journal Homepage
https://pub.scientificirg.com/index.php/ESIT

Intelligent MITM Attack Detection Systems Using Ensemble Learning for IoT Network Security

Yasser AbdelSatar[a,1] **, Fatma Elzahraa Mohamed**[b]**, Shimaa AbdelNasser**[c]**, and Ayah Alaa Mohamed**[d]

[a] *Department of Artificial Intelligence, Faculty of Computers and Artificial Intelligence, Sphinx University, Assiut 71511, Egypt; E-mail: yasser.selim@sphinx.edu.eg*

[b] *Department of Computer Science, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt; E-mail: fatma_mohamed@adj.aast.edu.eg*

[c] *Department of Computer Science, Assiut University, Assiut, Egypt; E-mail: shimaansr1@gmail.com*

[d] *Department of Computer Science, Arab Academy for Science, Technology and Maritime Transport, Aswan, Egypt; E-mail: ayahalaa52@gmail.com*

ABSTRACT

The expansion of Internet of Things (IoT) deployments has widened the attack surface available to adversaries, and the man-in-the-middle (MITM) attack remains one of the most damaging threats facing these networks. In a MITM attack, two parties that believe they are communicating directly are in fact exchanging traffic through an intermediary that silently alters or observes the exchange. Common realizations of this threat include ARP spoofing, DNS hijacking, and SSL stripping, each producing a distinct signature in network traffic that a classifier can learn to recognize. This study evaluates and compares five ensemble learning algorithms, Random Forest, Extra Trees, XGBoost, CatBoost, and LightGBM, for the detection of MITM activity in the TON_IoT network traffic dataset. Performance is assessed using accuracy, precision, recall, F1-score, area under the ROC curve, and computational cost. CatBoost obtained the highest detection accuracy (99.2%) and F1-score (0.987), while LightGBM required roughly one third of CatBoost's training time at a negligible cost in detection quality. Across all five algorithms, boosting methods showed a small but consistent advantage over bagging methods, and detection was effective for every MITM technique considered, with SSL stripping proving the most difficult to identify. The results suggest that the choice among these algorithms in an operational deployment should depend on whether the priority is raw detection accuracy, inference speed, or interpretability, rather than on accuracy alone.

PAPER INFORMATION

HISTORY

Received: 15 March 2026
Revised: 13 May 2026
Accepted: 19 June 2026
Online: 29 June 2026

MSC

68T07; 68T09; 94A12; 68M10; 94A08

KEYWORDS

Man-in-the-Middle;
Attack Detection;
Ensemble Learning;
TON_IoT.

1. INTRODUCTION

The rapid expansion of the Internet of Things (IoT) across industrial, medical, and residential settings has created new opportunities for efficiency gains, but it has also introduced significant security exposure. Most IoT devices are designed primarily for low cost and ease of use, with security treated as a secondary concern, which makes them an attractive foothold for adversaries seeking to reach larger networks [1].

[1]Corresponding author at Department of Artificial Intelligence, Faculty of Computers and Artificial Intelligence, Sphinx University, Assiut 71511, Egypt; E-mail: yasser.selim@sphinx.edu.eg.

Among the attack vectors available in these environments, the man-in-the-middle (MITM) attack is particularly damaging. By positioning itself between two communicating parties, an attacker can intercept credentials, inject malicious content, and redirect traffic toward a fraudulent destination, all while both legitimate endpoints believe they are communicating directly with one another [2]. Because of this stealthy positioning, conventional perimeter defenses are often unable to detect the attack while it is occurring.

Traditional signature-based intrusion detection struggles to keep pace with the variety of MITM techniques, since attackers continually adapt the specific mechanism used to insert themselves into a connection. Machine learning approaches mitigate this limitation by learning the statistical and temporal patterns present in traffic rather than relying on fixed signatures, which gives them a better chance of generalizing to attack variants that have not been seen before [3].

Ensemble learning, which combines the predictions of multiple base learners, has shown consistent advantages in cybersecurity applications because the combination of diverse models tends to reduce variance and improve generalization relative to a single classifier [4]. Even so, a systematic comparison of multiple ensemble families specifically for MITM detection in IoT traffic has been largely absent from the literature, a gap that recent work in this journal on related security and reliability problems, such as SQL injection detection and photovoltaic fault diagnosis, has begun to address from adjacent angles [5, 6].

This study compares five widely used ensemble algorithms on their ability to recognize the traffic signatures associated with three MITM techniques: ARP spoofing, DNS hijacking, and SSL stripping. The TON_IoT network traffic dataset [7] is used because it combines realistic, continuously generated background traffic with periodically injected attacks, which makes the detection task closer to an operational deployment than datasets built from isolated attack captures.

The contributions of this work are threefold:

1. A description of the traffic-level signatures associated with ARP spoofing, DNS hijacking, and SSL stripping, and how each signature is expected to manifest in flow-level features;
2. A controlled comparison of five ensemble algorithms, two bagging based and three gradient-boosting based, on the same dataset, split, and evaluation protocol; and
3. Operational guidance on which algorithm is preferable under different deployment constraints, including accuracy-critical, resource-constrained, and interpretability-critical settings.

The remainder of this paper is organized as follows. Section 2 reviews related work on ensemble learning for intrusion detection and on MITM detection specifically. Section 3 describes the three MITM techniques considered in this study and the traffic-level signatures they produce. Section 4 describes the dataset, preprocessing pipeline, and experimental design. Section 5 presents the results and discusses their operational implications. Section 6 concludes the paper and outlines directions for future work.

2. RELATED WORK

2.1 Ensemble Learning for Network Intrusion Detection

Ensemble methods owe their effectiveness in intrusion detection to their ability to reduce variance, bias, or both, by combining multiple base learners into a single decision rule [8, 4]. Breiman introduced Random Forest as a bagging method that trains many decision trees on bootstrapped samples while restricting the set of features considered at each split, producing trees that are individually noisy but collectively low in variance once their predictions are averaged [9]. Geurts et al. proposed Extra Trees, which adds a further layer of randomization by selecting split thresholds at random rather than searching for the locally optimal cut point, typically using the entire training set instead of a bootstrap sample; this additional randomness tends to lower variance further at the cost of a small increase in bias [10].

Gradient boosting methods take a different approach, building trees sequentially so that each new tree corrects the errors of the ensemble built so far. Chen and Guestrin's XGBoost popularized this approach by combining it with a regularized objective and a number of systems-level optimizations for speed and scalability [11]. Ke et al. introduced LightGBM, which uses gradient-based one-side sampling and exclusive feature bundling to reduce training cost on high-dimensional data without a substantial loss in accuracy [12]. Prokhorenkova et al. introduced CatBoost, which uses ordered boosting to reduce a form of target leakage known as prediction shift and provides native support for categorical features, removing the need for one-hot encoding of protocol and service fields that are common in network traffic data [13].

Several studies have benchmarked these algorithms on general-purpose intrusion detection datasets. Zhang et al. combined SMOTE oversampling with a convolutional architecture on the NSL-KDD dataset and reported gains over boosting baselines that were not similarly augmented for class imbalance [14]. On UNSW-NB15, Hasan et al. found that

CatBoost outperformed several alternative classifiers, including bagging methods, in both binary and multi-class attack detection [15]. Using the TON_IoT dataset specifically, Ahmed et al. demonstrated that a tuned gradient-boosting pipeline improved detection of a broad set of IoT network intrusions relative to conventional baselines, underscoring the value of boosting-based methods for this dataset family [16]. The same family of stacked and boosted ensemble methods has proven effective outside network security as well; for example, Ahmed et al. applied a stacked ensemble of pretrained convolutional networks and XGBoost to peripheral blood smear images for leukemia classification, reporting accuracy above 99% on an independent clinical dataset [17], which illustrates that the accuracy gains typically attributed to boosting in tabular network-traffic settings are not specific to that domain.

2.2 Detection of Man-in-the-Middle Attacks

Compared with the broader intrusion detection literature, work that targets MITM detection specifically is more limited and has tended to evaluate single classifiers rather than systematically comparing ensemble families. Saed and Aljuhani applied a Random Forest classifier to MITM traffic and reported strong detection performance, but did not compare against alternative ensemble methods [18]. Sultan et al. compared five classifiers, including XGBoost, K-nearest neighbors, and a multilayer perceptron, for MITM detection on MQTT traffic captured from IoT devices, and found that gradient boosting and tree-based methods were competitive with, and in several cases better than, the neural network baseline [19]. Michelena et al. proposed an intelligent detection scheme tailored to MQTT-based IoT environments and reported that hybrid models combining feature selection with classical classifiers outperformed single-model baselines on a synthetic MITM dataset [20].

Narang et al. surveyed the literature on MITM detection for the Internet of Medical Things and concluded that most published studies rely on a single machine learning model evaluated on a single dataset, which limits the generality of reported accuracy figures and motivates further benchmarking work across model families [21]. More recently, Fereidouni et al. reviewed MITM attacks across the layers of the IoT protocol stack, covering ARP spoofing, DNS manipulation, and SSL stripping among other techniques, and highlighted machine learning and blockchain as the two technologies most often proposed for mitigation, while noting that heterogeneous and resource-constrained IoT devices complicate deployment of either approach [22]. Ali and Al-Sharafi compared Random Forest, an LSTM network, and a support vector machine for MITM detection in IoT traffic and found that Random Forest achieved the highest accuracy among the three, though with comparatively low recall, illustrating that a single accuracy figure can mask weaknesses relevant to a security deployment [23].

Taken together, this body of work establishes that machine learning is effective for MITM detection in principle, but it leaves two questions comparatively underexplored: how a representative set of bagging and boosting ensembles compares once accuracy, recall, and computational cost are all considered together, and whether their relative performance differs across the distinct sub-techniques that fall under the MITM umbrella (ARP spoofing, DNS hijacking, and SSL stripping) rather than against MITM as a single undifferentiated class. The present study is designed to address both questions using a consistent dataset, split, and tuning protocol across five algorithms.

3. BACKGROUND: MAN-IN-THE-MIDDLE ATTACK TECHNIQUES

A MITM attack succeeds when both communicating parties believe they are exchanging data directly, while in fact every message passes through an attacker who can read, modify, or selectively drop it (**Figure 1**). The specific mechanism used to insert the attacker into the connection varies by protocol layer, and each mechanism leaves a different trace in network traffic, which is what ultimately makes the attack detectable by a classifier trained on flow-level features.

3.1 ARP Spoofing

ARP spoofing targets the Address Resolution Protocol, which maps IP addresses to MAC addresses within a local network segment. An attacker broadcasts forged ARP replies that bind its own MAC address to the IP address of a legitimate device, most commonly the default gateway. Once other hosts on the segment update their ARP caches with this forged mapping, traffic intended for the gateway is delivered to the attacker instead [24]. Because ARP operates without authentication, nothing in the protocol itself prevents this. In captured traffic, the attack typically appears as multiple ARP replies claiming the same IP address from different MAC addresses, or as ARP replies that arrive without a matching prior request, both of which are anomalous under normal protocol behavior.

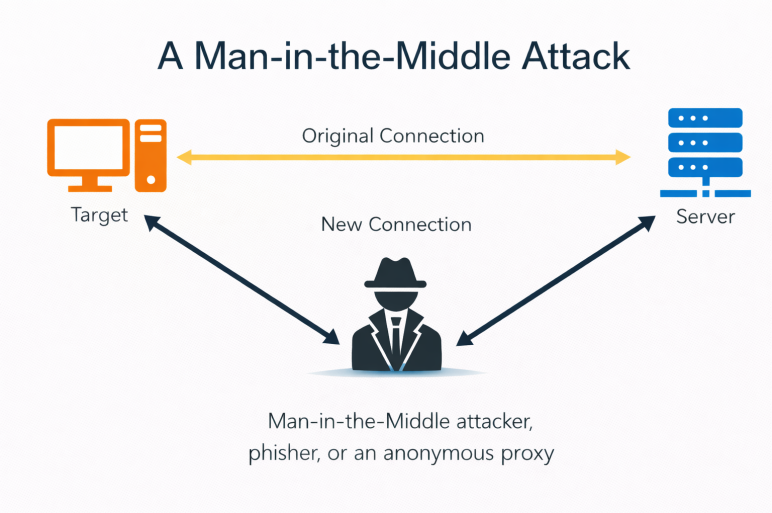

Figure 1: General structure of a man-in-the-middle attack: both endpoints believe they are exchanging traffic directly, while the attacker relays, observes, or alters every message

3.2 DNS Hijacking

DNS hijacking operates at a different layer, targeting the resolution of domain names to IP addresses rather than the resolution of IP addresses to MAC addresses (**Figure 2**). An attacker who can inject a forged DNS response, whether through cache poisoning, a rogue resolver, or control of an on-path router, causes a victim's query for a legitimate domain to resolve to an attacker-controlled IP address instead of the correct one. The victim's subsequent connection is then established with the attacker rather than the intended server, while the victim's client software has no indication that anything is wrong, since the DNS response appears syntactically valid [25]. In traffic, this technique tends to produce DNS responses with an unexpectedly short time-to-live value, an unusually large number of DNS queries or responses originating from a single source in a short window, or response packets whose source address does not match the network's configured authoritative resolver.

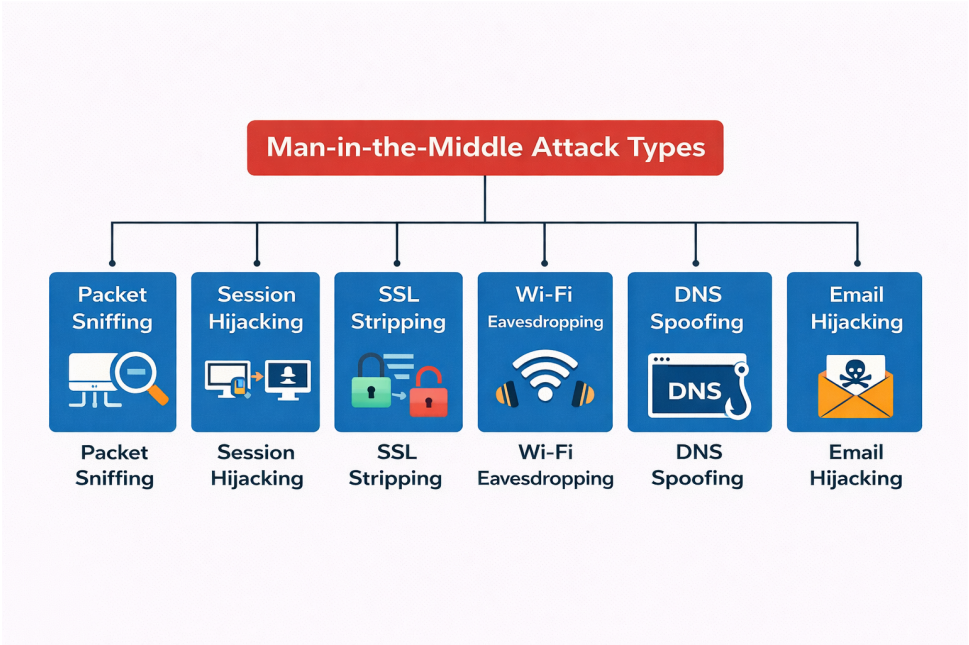

Figure 2: Common categories of man-in-the-middle techniques, of which ARP spoofing, DNS hijacking, and SSL stripping are the focus of this study

3.3 SSL Stripping

SSL stripping is more advanced than the two techniques above because it targets the negotiation of an encrypted connection rather than the routing of traffic that is already established. Instead of intercepting traffic after a secure channel has been negotiated, the attacker intercepts the initial request and forces the connection to downgrade from HTTPS to HTTP [26]. The victim's browser then communicates with the attacker over an unencrypted channel that the attacker can read and modify freely, while the attacker maintains a separate, legitimate HTTPS connection to the real server on the victim's behalf, so the round trip appears complete from the server's point of view. Because many users do not notice the absence of the lock icon or the change from HTTPS to HTTP in their browser, this downgrade is often the only externally visible sign of the attack. In traffic, the attack is detectable through HTTP responses appearing on ports conventionally reserved for HTTPS, and through protocol transition patterns in which a client that has previously used HTTPS with a given server unexpectedly falls back to plain HTTP.

4. METHODOLOGY

4.1 Dataset Description

This study uses the TON_IoT dataset, a benchmark security dataset collected at the UNSW Canberra Cyber Range and IoT Labs from a testbed that combines IoT and Industrial IoT devices, edge and fog gateways, and cloud services [7]. Unlike datasets that capture only a background period followed by a single isolated attack, TON_IoT was collected with normal traffic generated continuously while a range of attacks, including MITM activity, were injected periodically, which makes the resulting traffic distribution closer to what a deployed detector would encounter in practice.

This study uses the network traffic sub-dataset of TON_IoT, which records 44 attributes covering packet-level fields, flow-level statistics, and basic protocol metadata. The original dataset includes several attack categories beyond MITM, such as denial-of-service, scanning, ransomware, and injection attacks; these categories were removed so that the resulting task is a binary classification problem distinguishing normal traffic from the three MITM sub-techniques considered in this study (ARP spoofing, DNS hijacking, and SSL stripping).

4.2 Data Preprocessing

Features with more than 30% missing values were removed; the remaining missing numerical values were imputed with the feature mean. Categorical fields, primarily protocol type and service indicators, required different treatment depending on the target algorithm: one-hot encoding was applied for algorithms without native categorical support (Random Forest, Extra Trees, XGBoost, LightGBM), while CatBoost was given the original categorical indicators directly, consistent with its native handling of this feature type [13].

Tree-based ensembles are insensitive to monotonic transformations of individual features, so feature scaling has no effect on the resulting splits. Numerical features were nonetheless normalized for consistency across algorithms and to support visual inspection of the data during exploratory analysis; this step did not materially change model behavior.

Class imbalance was a substantial concern: MITM instances made up a small minority of the dataset after the non-MITM attack categories were removed, consistent with the rarity of MITM activity relative to background traffic in the underlying testbed.

4.3 Experimental Design

The dataset was partitioned using stratified sampling into 80% training and 20% testing sets, preserving the original class ratio in both partitions. Hyperparameters for each algorithm were tuned with Bayesian optimization over 100 iterations, using search ranges drawn from the literature and preliminary exploratory runs. For Random Forest and Extra Trees, the number of trees, maximum depth, and minimum samples required for a split or a leaf were tuned. For the three boosting algorithms, the learning rate, regularization strength, and algorithm-specific parameters (for example, the number of leaves for LightGBM and the depth and L2 regularization coefficient for CatBoost) were tuned in addition to the tree-structure parameters shared with the bagging methods.

Five-fold cross-validation was used during the tuning process described above. The held-out test set, set aside before any tuning began, was used only once, to compute the final performance metrics.

4.4 Evaluation Metrics

Because MITM instances are a small minority of the dataset, accuracy alone is not an adequate metric: a trivial classifier that labels every instance as normal traffic would already achieve a high accuracy score while detecting no attacks at all. For this reason, the following metrics are reported in addition to accuracy, using true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) defined relative to the MITM class:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Precision (**Equation 1**) measures the proportion of instances flagged as MITM that are genuine attacks, which is the quantity that determines the false-alarm burden placed on a security operator. Recall (**Equation 2**) measures the proportion of genuine MITM instances that the model successfully flags, which determines how many real attacks go undetected. The F1-score (**Equation 3**) is the harmonic mean of precision and recall and is reported as a single summary statistic when both quantities need to be considered together.

The area under the receiver operating characteristic curve (AUC-ROC) is also reported. The ROC curve plots the true positive rate against the false positive rate as the classification threshold is varied, and the area under this curve summarizes a classifier's ability to separate the two classes independently of any single operating threshold; a value of 1.0 indicates perfect separation, and a value of 0.5 indicates performance equivalent to random guessing.

Finally, training time (the wall-clock time required to fit a model on the training set) and inference time (the wall-clock time required to classify a single new instance) are reported, since both quantities determine whether a given algorithm is practical for a specific deployment, for example frequent retraining on resource-constrained IoT gateways.

5. RESULTS AND DISCUSSION

5.1 Detection Performance

Table 1 compares the five algorithms across all reported metrics. CatBoost obtained the highest F1-score (0.987), closely followed by XGBoost (0.982) and LightGBM (0.982). Random Forest and Extra Trees were competitive with the boosting methods but trailed them by a small margin on every metric except training time.

Table 1: Performance comparison across the five ensemble algorithms on the held-out TON_IoT test set

Algorithm	Accuracy	Precision	Recall	F1-Score	AUC-ROC	Train Time (s)
Random Forest	0.987	0.974	0.981	0.977	0.992	124.3
Extra Trees	0.986	0.971	0.979	0.975	0.991	98.7
XGBoost	0.991	0.984	0.980	0.982	0.996	156.2
CatBoost	0.992	0.989	0.985	0.987	0.997	203.8
LightGBM	0.990	0.982	0.983	0.982	0.995	67.5

A McNemar's test comparing CatBoost against each bagging method (Random Forest and Extra Trees) found the difference in misclassification patterns to be statistically significant ($p < 0.01$), following the testing procedure described by Java et al. for comparing intrusion detection classifiers [27]. The same test applied between CatBoost and the other two boosting methods (XGBoost and LightGBM) did not reach significance, indicating that the gap among the three boosting algorithms is small enough that it cannot be distinguished reliably from chance variation on this test set.

5.2 Attack-Type Analysis

Detection performance differed across the three MITM sub-techniques. ARP spoofing was identified with high recall by every algorithm, which is consistent with the fact that ARP spoofing operates at a fixed, low-level protocol layer with relatively little room for an attacker to vary its implementation. SSL stripping was the most difficult of the three techniques

to detect, with recall 2 to 3 percentage points lower than for ARP spoofing across all five algorithms; this is consistent with SSL stripping operating at the application layer, where its traffic pattern can more easily resemble ordinary unencrypted HTTP usage. DNS hijacking fell between these two extremes, reflecting its position at an intermediate protocol layer.

5.3 Computational Efficiency

Training time differences across the five algorithms followed their expected algorithmic complexity. Random Forest and Extra Trees trained faster than the boosting methods owing to the embarrassingly parallel nature of building independent trees. Among the three boosting algorithms, LightGBM trained roughly three times faster than CatBoost, a direct consequence of its histogram-based, gradient-based one-side sampling strategy [12].

Inference time showed much less variation across algorithms. On the test hardware used in this study (Intel Xeon E5-2680, 64 GB RAM), every algorithm classified a new instance in under 5 milliseconds, with Extra Trees fastest and CatBoost slowest; even the slowest algorithm remained well within the latency budget of most real-time network monitoring deployments.

5.4 Feature Importance Analysis

Table 2 reports feature importance scores derived from the trained Random Forest model, which is used here for interpretability rather than as the best-performing algorithm.

Table 2: Feature importance scores from the trained Random Forest model

Network Feature	Relative Importance
src_ip	0.175
src_ip_bytes	0.105
src_port	0.080
conn_state	0.075
dst_port	0.070
dst_ip	0.065
dst_ip_bytes	0.060
duration	0.050
src_pkts	0.045
src_bytes	0.040
dst_pkts	0.025
dst_bytes	0.020
proto	0.015
dns_rejected	0.010
service	0.005
dns_RA	0.002
dns_qtype	0.001
dns_query	-0.001
dns_rcode	-0.002
dns_AA	-0.003

The most influential features, source IP, source IP bytes, source port, connection state, and destination port, are consistent with the mechanics of a MITM attack, which by construction involves a specific set of hosts and alters the volume or pattern of traffic flowing through them. A number of DNS-related fields received slightly negative importance scores, indicating that including them provided no net benefit and may have introduced noise; this is plausible given that DNS hijacking is only one of the three MITM sub-techniques represented in the dataset, so DNS-specific fields carry no signal for the ARP spoofing and SSL stripping instances.

Across all five algorithms, four feature categories were consistently relevant:

- **ARP request-to-response ratios**: an elevated ratio is characteristic of spoofing attempts, since a spoofing host typically issues more unsolicited replies than a normal host would.
- **DNS response time-to-live values**: unusually short TTL values are characteristic of hijacking, since attacker-controlled records are often configured to expire quickly to support rapid redirection.

- **Protocol transition counts**: an HTTP connection immediately following prior HTTPS use to the same destination is characteristic of SSL stripping.
- **Inter-arrival timing irregularities**: MITM relaying introduces processing delay that produces a measurably different distribution of packet inter-arrival times compared with direct client-server communication.

Timing-related features were generally more informative than simple count or size-based features, which suggests that the latency introduced by relaying traffic through an attacker is harder to disguise than the volume or size of that traffic.

5.5 Implications for Deployment

The high performance achieved by all five algorithms indicates that ensemble methods are broadly effective for MITM detection in IoT environments, but the choice among them should depend on more than the overall accuracy figure in**Table 1**.

In resource-constrained environments, such as IoT monitoring deployed on gateway hardware with limited compute budget, LightGBM is the more practical choice. Its training time supports frequent retraining as new attack variants are observed, and its inference time is low enough for deployment on modest hardware, at a cost in F1-score relative to CatBoost that is within the range that the significance test could not distinguish from chance.

In high-security environments where the cost of a missed detection is high, the additional computational cost of CatBoost may be justified by its slightly higher recall. A 0.5 percentage point improvement in recall relative to LightGBM corresponds to roughly 110 additional detected attacks on a test set of this size, which is a meaningful difference when protecting critical infrastructure.

When interpretability is a requirement, for example to support a security analyst's investigation or a compliance audit, Random Forest and Extra Trees remain attractive despite their slightly lower accuracy, because their individual decision paths can be inspected directly, whereas the boosting methods require a separate post-hoc explanation technique to achieve comparable transparency.

5.6 Limitations

This study has several limitations. TON_IoT, while a current and widely used benchmark for IoT network security research, was collected in a controlled testbed, and laboratory traffic is generally less varied and more predictable than traffic on an operational network; performance on TON_IoT should therefore be treated as an upper bound rather than a guarantee of equivalent performance in the field.

This study also treats the three MITM sub-techniques as a single binary class (MITM versus normal) for the purpose of training and primary evaluation. A multi-class formulation that distinguishes ARP spoofing, DNS hijacking, and SSL stripping during training, rather than only during post-hoc analysis, is a natural extension and may reveal additional differences in feature relevance across sub-techniques.

6. CONCLUSION AND FUTURE WORK

This study compared five ensemble learning algorithms, Random Forest, Extra Trees, XGBoost, CatBoost, and LightGBM, for the detection of man-in-the-middle traffic in IoT networks using the TON_IoT dataset. All five algorithms achieved strong detection performance, with the gradient-boosting methods showing a small but statistically significant advantage over the bagging methods. CatBoost achieved the highest overall detection quality, LightGBM offered the best balance between accuracy and computational cost, and Random Forest and Extra Trees remained the most interpretable choices for analysts who need to inspect individual decisions.

Detection was effective for all three MITM sub-techniques considered, with ARP spoofing the easiest to detect because of its fixed, low-level protocol signature, and SSL stripping the most difficult because of its closer resemblance to ordinary application-layer traffic.

Future work should pursue a multi-class formulation that distinguishes between MITM sub-techniques during training, evaluate these algorithms under adversarial conditions designed to evade the specific traffic features, and assess their performance when deployed on representative edge hardware rather than evaluated offline on held-out test data.

AUTHOR CONTRIBUTION STATEMENT

All authors contributed equally to the study conception and design. Material preparation, data collection, and analysis were performed by the authors. The first draft of the manuscript was written by the authors, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

This study did not involve human participants or animals; ethical approval and consent to participate are therefore not applicable.

CONSENT FOR PUBLICATION

Not applicable.

DATA AVAILABILITY

The TON_IoT dataset used in this study is publicly available from the UNSW Canberra Cyber Range repository at `https://research.unsw.edu.au/projects/toniot-datasets`.

ACKNOWLEDGMENTS

The authors thank the reviewers, the Associate Editor, and the Editor-in-Chief for their valuable comments and suggestions, which improved the quality of this paper.

FUNDING

No Funding.

DISCLOSURE STATEMENT

The authors declare that they have no competing interests.

REFERENCES

[1] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of network and computer applications*, vol. 88, pp. 10–28, 2017.

[2] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[4] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2025.

[5] Y. A. Satar, A. Mohamed, and A. A. Hassanain, "Nsga-ii optimization of probabilistic neural networks for robust sql injection attack detection," *Engineering Systems and Intelligent Technologies (ESIT)*, vol. 1, no. 1, pp. 29–41, 2026.

[6] Y. A. Satar, H. A. Almansouri, and A. A. Hassanain, "Cvar-optimized distributionally robust stacked ensemble with range-based current signatures for reliable fault detection in photovoltaic farms," *Engineering Systems and Intelligent Technologies (ESIT)*, vol. 1, no. 1, pp. 1–14, 2026.

[7] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.

[8] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, pp. 1–15, Springer, 2000.

[9] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[12] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[13] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[14] H. Zhang, L. Huang, C. Q. Wu, and Z. Li, "An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset," *Computer Networks*, vol. 177, p. 107315, 2020.

[15] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things*, vol. 7, p. 100059, 2019.

[16] M. A. O. Ahmed, Y. Abdelsatar, R. Alotaibi, and O. Reyad, "Enhancing internet of things security using performance gradient boosting for network intrusion detection systems," *Alexandria Engineering Journal*, vol. 116, pp. 472–482, 2025.

[17] M. A. Ahmed, R. Alotaibi, Y. A. Satar, N. Gaber, N. F. Omran, and O. Reyad, "Fast detection of acute lymphoblastic leukemia through stacked pre-trained ensemble learning and efficient segmentation," *Arabian Journal for Science and Engineering*, pp. 1–14, 2025.

[18] M. Saed and A. Aljuhani, "Detection of man in the middle attack using machine learning," in *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, pp. 388–393, IEEE, 2022.

[19] A. B. M. Sultan, S. Mehmood, and H. Zahid, "Man in the middle attack detection for mqtt based iot devices using different machine learning algorithms," in *2022 2nd international conference on artificial intelligence (ICAI)*, pp. 118–121, IEEE, 2022.

[20] Á. Michelena, J. Aveleira-Mata, E. Jove, M. Bayón-Gutiérrez, P. Novais, O. F. Romero, J. L. Calvo-Rolle, and H. Aláiz-Moretón, "A novel intelligent approach for man-in-the-middle attacks detection over internet of things environments based on message queuing telemetry transport," *Expert Systems*, vol. 41, no. 2, p. e13263, 2024.

[21] M. Narang, A. Jatain, and N. Punetha, "A survey on detection of man-in-the-middle attack in iomt using machine learning techniques," in *International Conference on Computational Intelligence*, pp. 117–132, Springer, 2023.

[22] H. Fereidouni, O. Fadeitcheva, and M. Zalai, "Iot and man-in-the-middle attacks," *Security and Privacy*, vol. 8, no. 2, p. e70016, 2025.

[23] M. A. Ali and S. A. H. Al-Sharafi, "Intrusion detection in iot networks using machine learning and deep learning approaches for mitm attack mitigation," *Discover Internet of Things*, vol. 5, no. 1, p. 48, 2025.

[24] S. Whalen, "An introduction to arp spoofing," *Node99 [Online Document]*, vol. 563, 2001.

[25] S. Son and V. Shmatikov, "The hitchhiker's guide to dns cache poisoning," in *International Conference on Security and Privacy in Communication Systems*, pp. 466–483, Springer, 2010.

[26] M. Marlinspike, "New tricks for defeating ssl in practice," *Black Hat DC*, vol. 2, 2009.

[27] M. I. Java, U. I. Shabrina, R. N. Fahmi, B. A. Pratomo, *et al.*, "Enhancing cybersecurity: Two-phase detection approach for intrusion network for anomaly data," in *2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, pp. 1–6, IEEE, 2024.