

Deep Learning for Enhancing IoT Security and Trust

Alireza Rahimipour Anaraki,

Department of Computer Engineering, CT.C., Islamic Azad University, Tehran, Iran
a.rahimipouranaraki@iau.ir

Article history: Received 7 November 2024, Revised 17 January, 2025, Accepted 3 February 2025.

Abstract—The rapid expansion of the Internet of Things has introduced large-scale security and trust challenges due to device heterogeneity, limited resources, and exposure to evolving cyberattacks. Traditional rule-based protection mechanisms and static trust policies often underperform against adaptive attackers and distributed threats. This paper presents a lightweight deep-learning-driven security and trust enhancement framework that combines sequence-based behavioral modeling with an adaptive trust controller to reduce false alarms and enable more reliable device-level decision making. The framework relies on a long short-term memory network for temporal traffic modeling, while continuously updating device trust using anomaly severity, historical consistency, and device criticality. A reproducible preprocessing pipeline is specified to transform heterogeneous network-flow features into fixed-length sequences. The paper also defines an evaluation protocol on widely used intrusion datasets and modern Internet-of-Things-oriented corpora, and reports a structured template for presenting detection performance and edge feasibility metrics, including parameter footprint, inference latency, memory usage, and energy consumption.

Keywords: IoT Security, Trust Management, Deep Learning, Intrusion Detection, Long Short-Term Memory, Convolutional Neural Network, Graph Neural Network.

1. Introduction

The Internet of Things (IoT) has emerged as a transformative paradigm of modern computing, connecting billions of heterogeneous devices spanning consumer, industrial, and critical infrastructure domains. As these devices sense, exchange, and act on physical-world data, the attack surface expands substantially and becomes difficult to defend with conventional perimeter-centric approaches.

IoT networks are targeted by a wide spectrum of attacks, including unauthorized access, malware injection, eavesdropping, spoofing, Sybil behavior, and routing manipulation. These threats are amplified by constrained compute, memory, and energy budgets, as well as lightweight protocols and dynamic topologies.

Deep learning has recently shown strong capability in modeling complex patterns in high-dimensional security data. Unlike traditional machine learning, deep models can learn feature representations directly from raw or lightly processed telemetry and can generalize across diverse traffic patterns. In parallel, trust management has been investigated as a complementary mechanism that assigns a trust score to devices and uses this score to guide actions such as throttling, isolation, or access restriction.

This paper focuses on improving IoT security by integrating deep-learning-based anomaly detection with an adaptive trust mechanism. The key novelty is a lightweight design intended for edge deployment, while explicitly encoding local structural context through feature engineering and role-aware trust updates.

Cite this article: Alireza Rahimipour Anaraki. Deep Learning for Enhancing IoT Security and Trust. *Journal of Artificial Intelligence Tools in Software and Data Engineering (AITSD)*, 3(1), pages 9-15.



The remainder of this paper is organized as follows: Section 2 reviews the theoretical foundations of IoT security, trust management, and deep-learning-based intrusion detection. Section 3 surveys related literature, with emphasis on recent hybrid and trust-aware frameworks. Section 4 presents the proposed lightweight LSTM-based detection and adaptive trust integration framework, including reproducible preprocessing and role-aware trust updates. Section 5 describes the performance evaluation methodology, datasets, baselines, and deployment measurements. Section 6 provides a structured Results and Discussion section with tables and figure placeholders for reporting empirical findings from the implementation. Finally, Section 7 concludes the paper.

2. Theoretical Background

2.1. IoT security and threat model

IoT security requires defending device endpoints, wireless links, and routing/coordination layers. Common attack classes include identity spoofing, replay, botnet recruitment, denial of service, data exfiltration, and topology-aware threats such as Sybil nodes or routing rank manipulation in low-power routing protocols. These threats are challenging because IoT traffic is heterogeneous, often bursty, and affected by mobility and intermittent connectivity.

2.2. trust management fundamentals

Trust management complements intrusion detection by representing device reliability as a continuously updated score. A trust score can be computed from behavioral consistency, protocol compliance, historical interactions, and security alerts. Dynamic trust enables graded responses, for example, temporarily limiting a device's privileges rather than immediately removing it from the network.

2.3. Deep learning for intrusion detection in IoT

Deep learning approaches used for intrusion detection include convolutional neural networks for learning spatial feature interactions, recurrent neural networks such as long short-term memory (LSTM) for modeling temporal dependencies, and graph neural networks for capturing topological relationships among communicating entities. Graph-based approaches can be particularly effective for coordinated and topology-dependent attacks, but may introduce additional cost due to graph construction and message passing.

3. Related Work

Lightweight temporal models have been widely explored for anomaly detection where streaming behavior is informative. Elsayed et al. proposed an LSTM-based autoencoder for network anomaly detection with emphasis on reconstructing normal patterns and identifying deviations [6]. Hybrid models that combine convolutional and recurrent modules can capture both feature interactions and temporal structure, and have been applied to IoT and industrial IoT scenarios [13].

Trust-aware security frameworks integrate anomaly scores with a controller that updates device trust and triggers adaptive access decisions. Ullah et al. proposed DeepTrust, a deep-learning-driven trust and reputation framework for IoT networks [4]. Naik et al. presented a hybrid trust-aware controller integrated with multiple deep modules and evaluated edge feasibility on common embedded platforms [2].

Recent graph-based intrusion detection models encode communication structure explicitly. For example, Lin et al. proposed a graph-neural-network-based approach with attention and contrastive learning to better capture topology-based intrusion patterns [12]. While effective, such methods can increase runtime and memory requirements due to neighborhood sampling and graph embedding operations.

3.1. Technical novelty and design rationale

In contrast to recent hybrid frameworks, this work intentionally adopts a single lightweight LSTM module to reduce model size and simplify deployment in resource-constrained environments. To address the structural limitation of purely temporal modeling, the framework explicitly injects local structural context into the LSTM input through feature engineering (Section 4.2) and uses a role-aware trust update to amplify security sensitivity for mission-critical nodes (Section 4.4). This design aims to provide a favorable trade-off between detection performance, false-positive reduction, and edge feasibility.

4. Proposed Framework

4.1. System overview

The proposed framework consists of (i) a lightweight LSTM-based anomaly detector operating on fixed-length sequences of traffic features and (ii) an adaptive trust module that updates each node's trust score based on anomaly severity and device context. Decisions (e.g., allow, throttle, isolate, alert) are triggered by trust thresholds and can be integrated into an access-control or routing policy.

4.2. Justification for a purely temporal LSTM and structural encoding

Although graph neural networks can directly model network structure, they typically require explicit graph construction and message passing, which may not be feasible on constrained edge devices or in highly dynamic IoT deployments. A single LSTM provides efficient temporal modeling with a compact parameter footprint. To mitigate structural blindness, the LSTM input is augmented with topology-aware engineered features that encode local structural context without full graph inference. Examples include: number of distinct peers per window, neighbor churn rate, hop count and routing rank (for RPL-like settings), link-quality indicators, retransmission ratios, and neighbor-consistency scores computed from trust-weighted interactions. These features enable the temporal model to learn signatures associated with spoofing, Sybil behavior, and routing inconsistencies.

4.3. Sequence construction and preprocessing for LSTM input

Most intrusion datasets provide heterogeneous tabular flow features rather than ready-to-use sequences. To construct sequences, records are grouped by device identifier (e.g., source IP/MAC or sensor ID), sorted by timestamp, and segmented using a sliding window of length W with stride S . Each timestep corresponds to one flow record (or an aggregated flow statistic), resulting in an input tensor of shape (N, W, F) , where N is the number of sequences and F is the final feature dimension after encoding.

Numerical features are normalized (e.g., z-score or min-max scaling). Categorical features (e.g., protocol, service, flag) are converted using one-hot encoding or learned embeddings. Time-derived features (e.g., inter-arrival time, burstiness) are computed per window. The window definition (W, S) and the exact set of engineered features must be reported to ensure reproducibility.

4.4. Adaptive trust update with severity and device role

Each device n is assigned a trust score $T_n(t)$ in $[0, 1]$. Trust is updated using a continuous anomaly severity score $s_n(t)$ in $[0, 1]$ rather than a binary anomaly flag. Let $r(n)$ denote the role of device n (e.g., sensor, actuator, gateway). Role-aware punishment and reward factors, α_r and β_r , control the trust dynamics:

$$T_n(t+1) = \text{clip}(T_n(t) + \beta_r\{r(n)\} \cdot (1 - s_n(t)) - \alpha_r\{r(n)\} \cdot s_n(t), 0, 1).$$

Mission-critical nodes (e.g., gateways) should use a larger punishment factor α and/or stricter trust threshold to ensure faster mitigation upon suspicious behavior, while low-impact sensors may use milder settings to reduce unnecessary disruptions.

4.5. Stability against adaptive attackers

A threshold-based detector can be bypassed by an attacker who remains marginally below the classification boundary. The severity-aware trust update accumulates small penalties over time, thereby reducing trust even when anomaly probabilities stay just below the detection threshold. Optionally, an exponential moving average of severity can penalize sustained borderline behavior while discounting isolated spikes.

4.6. Parameter optimization of α and β

The parameters α_r and β_r should be selected through validation-driven optimization under heterogeneous conditions. A practical approach is to perform grid search or Bayesian optimization over α_r and β_r to minimize a weighted objective that penalizes false positives, false negatives, and instability (rapid oscillations of trust). The chosen values and optimization procedure must be reported (Section 5.5) for reproducibility.

4.7. Pseudocode of the proposed framework

Algorithm 1. Trust-Aware Lightweight LSTM Intrusion Detection

Inputs: Sequence features $x(t)$, anomaly severity $s(t)$ in $[0, 1]$, device role r , role-aware parameters α_r and β_r , trust threshold τ .

Outputs: Updated trust score $T(t)$, mitigation action $A(t)$.

```

Initialize  $T(0) = 0.5$  for each device
For each time window  $t$  do
     $y(t) = \text{LSTM}(x(t))$                                 # anomaly
probability
     $s(t) = \text{severity}(y(t))$                             # continuous
severity in  $[0, 1]$ 
     $\text{penalty} = \alpha_r * s(t)$ 
     $\text{reward} = \beta_r * (1 - s(t))$ 
     $T(t+1) = \text{clip}(T(t) + \text{reward} - \text{penalty}, 0, 1)$ 
    If  $T(t+1) < \tau$  then
         $A(t) = \{\text{isolate} / \text{throttle} / \text{alert}\}$ 
    Else
         $A(t) = \{\text{allow}\}$ 
    End if
End for

```

4.8. Lightweightness analysis

To substantiate the lightweight claim, the model size and memory footprint must be reported. For a single LSTM layer with input dimension F and hidden size H , the number of trainable parameters is:

$$P_{\text{LSTM}} = 4 \cdot (H \cdot (H + F) + H).$$

The approximate memory footprint in bytes is $P_{\text{LSTM}} \cdot b$, where b is the bytes per parameter (e.g., 4 bytes for float32, 1 byte for int8). In Section 6, the paper reports parameter count and memory for the proposed model and compares them against hybrid baselines such as Naik et al. [2].

4.9. Performance Evaluation

Data-entry checklist for final submission: (i) fill Table 1 metrics (Accuracy/F1/FPR) for each dataset; (ii) report trust-module FPR reduction; (iii) report edge deployment metrics (latency, RAM, power) from your device; and (iv) confirm W, S, and F for reproducibility.

5. Implementation details

The framework is implemented in Python using a standard deep learning library. Training is performed on a workstation GPU/CPU, while inference measurements (latency, RAM, and power) are collected on an edge platform.

Edge platform (specify model and configuration):

Edge hardware (model/CPU/RAM/OS) used for latency/RAM/power measurements: Raspberry Pi 4 Model B (4GB), ARM Cortex-A72 1.5 GHz, Raspberry Pi OS 64-bit.

5.1. Datasets

To address reviewer concerns regarding IoT realism, experiments are conducted on both traditional IDS benchmarks and IoT-oriented datasets: NSL-KDD [5] and CICIDS2017 [7] provide widely used baseline comparisons, while Bot-IoT [8] and TON-IoT [9] better reflect modern IoT attack patterns and telemetry characteristics.

In addition, protocol-specific features relevant to lightweight IoT settings, such as RPL routing indicators and CoAP interaction statistics, are included when available or derived through feature engineering [10,11].

5.2. Sequence construction settings

Window length W (timesteps):

Sequence window length W (timesteps per sample): 20.

Stride S :

Sliding step size S (stride): 5.

Final feature dimension F after encoding:

Final feature dimension F after encoding (report per dataset): 122 (NSL-KDD); 78 (CICIDS2017); 115 (Bot-IoT); 125 (TON-IoT) — after encoding.

5.3. Baselines and comparison models

Baselines should include classical machine learning models (e.g., logistic regression, support vector machines, random forests, gradient-boosted trees) and representative deep learning models. For comparison with hybrid state-of-the-art, Naik et al.'s trust-aware hybrid framework is used as a reference model [2].

5.4. Metrics and reporting requirements

Detection performance is reported using Accuracy, Precision, Recall, F1-score, and False Positive Rate (FPR). To demonstrate the trust module's effectiveness, FPR is reported for (i) the standalone LSTM detector and (ii) LSTM combined with the trust module. Edge feasibility is quantified via inference latency (ms), RAM footprint (MB), and power consumption (mW) under a fixed batch size and input window.

5.5. Hyperparameter selection for α and β

The role-aware trust parameters α_r and β_r are tuned on a validation set. A recommended procedure is to search α_r in $[0.01, 0.5]$ and β_r in $[0.01, 0.5]$ while holding τ fixed (or jointly tuning τ), and to select values that minimize FPR subject to a minimum recall constraint. The final selected values for each role must be reported.

Selected α_r and β_r values (report per device role):

Role-aware trust parameters (α_r , β_r) for each device role: Gateway: $\alpha=0.12$, $\beta=0.03$; Actuator: $\alpha=0.08$, $\beta=0.04$; Sensor: $\alpha=0.06$, $\beta=0.05$; Trust threshold $\tau=0.35$.

6. Results and Discussion

Ethics and reproducibility note: All numeric results (Accuracy/F1/FPR and edge latency/RAM/power) must be obtained from actual experiments. This manuscript template intentionally leaves these fields as 'TBD (measured)' to avoid presenting fabricated results.

This section reports empirical findings obtained from executing the proposed framework. To comply with the review requirements, tables below must be populated with measured results from the author's implementation, and corresponding plots (e.g., ROC, PR, confusion matrices) should be inserted.

Table 1. Detection performance comparison across datasets and baselines.

Dataset	Model	Accuracy	F1-score	FPR	Notes
NSL-KDD [5]	LSTM (ours)	TBD	TBD	TBD	Enter your measured results (do not use estimates).
NSL-KDD [5]	Random Forest	TBD	TBD	TBD	
CICIDS2017 [7]	LSTM (ours)	TBD	TBD	TBD	
CICIDS2017 [7]	XGBoost	TBD	TBD	TBD	
Bot-IoT [8]	LSTM (ours)	TBD	TBD	TBD	IoT-oriented dataset
TON-IoT [9]	LSTM (ours)	TBD	TBD	TBD	IoT/IIoT telemetry

Table 2. Effect of the trust module on false positives (lower is better).

Dataset	FPR (LSTM only)	FPR (LSTM + Trust)	Reduction (%)
NSL-KDD [5]	TBD	TBD	TBD
CICIDS2017 [7]	TBD	TBD	TBD
Bot-IoT [8]	TBD	TBD	TBD
TON-IoT [9]	TBD	TBD	TBD

Table 3. Model lightweights comparison (parameters and memory).

Model	Trainable params	Memory (float32, MB)	Memory (int8, MB)	Comment
LSTM (ours)	47,872	0.18	0.05	Example configuration: H=64, F=122
Naik et al. hybrid [2]	Report from cited paper	Report from cited paper	Report from cited paper	Fill using the referenced hybrid model description

Table 4. Edge feasibility measurements.

Edge hardware	Latency (ms)	RAM footprint (MB)	Power (mW)	Measurement notes
Raspberry Pi 4 Model B (4GB), ARM Cortex-A72 1.5GHz, Raspberry Pi OS 64-bit	TBD (measured)	TBD (measured)	TBD (measured)	Batch=1, float32; replace with your measurement settings

Figure 1 (placeholder). Framework overview diagram to be inserted (data flow: preprocessing → LSTM detector → severity → trust update → action).

Figure 2 (placeholder). ROC/PR curves comparing LSTM-only vs LSTM+Trust (insert plot generated from your experiments).

Figure 3 (placeholder). Trust score trajectories under adaptive attacker behavior (insert plot demonstrating stability).

6.1. Discussion and compliance with reviewer requirements

The results should be discussed with emphasis on: (i) comparative performance against baselines, (ii) the trust module's contribution to lowering FPR without harming recall, (iii) robustness against adaptive attackers who attempt to remain just below detection thresholds, and (iv) deployment feasibility on edge hardware. In particular, reviewers requested hard evidence for lightweights; thus, parameter count and memory footprint (Table 3) and on-device latency/power/RAM (Table 4) are mandatory.

7. Conclusion

This paper presented a lightweight, trust-aware deep learning framework for IoT security that combines an LSTM-based anomaly detector with an adaptive trust controller updated by anomaly severity and device role. To overcome the structural limitations of purely temporal modeling, the framework injects local topology-aware context through engineered features and applies role-aware punishment and reward factors to prioritize mission-critical nodes. The paper also provided a reproducible sequence construction pipeline and a rigorous evaluation protocol covering both traditional benchmarks and IoT-oriented datasets, alongside mandatory edge feasibility measurements (latency, RAM, and power). Future work will focus on incorporating explicit graph learning when sufficient edge resources are available, extending the trust model with richer context and federated learning, and validating the framework under additional real-world IoT deployments.

8. References

- [1] I. Ahmed, I. Yaqoob, I. A. T. Hashem, A. Ahmed, and M. Imran, "Internet of Things security: A survey and future research directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 909-955, 2021. doi: 10.1109/COMST.2021.3054410.
- [2] N. Naik, N. Surendranath, S. A. B. Raju, C. Madduri, N. Dasari, V. K. Shukla, and V. Patil, "Hybrid deep learning-enabled framework for enhancing security, data integrity, and operational performance in Healthcare Internet of Things (H-IoT) environments," *Scientific Reports*, vol. 15, 31039, 2025. doi: 10.1038/s41598-025-15292-2.
- [3] F. S. Alsubaei, "Smart deep learning model for enhanced IoT intrusion detection," *Scientific Reports*, vol. 15, 20577, 2025. doi: 10.1038/s41598-025-06363-5.
- [4] F. Ullah, A. Salam, F. Amin, I. A. Khan, J. Ahmed, S. A. Zaib, and G. S. Choi, "Deep Trust: A Novel Framework for Dynamic Trust and Reputation Management in the Internet of Things (IoT)-Based Networks," *IEEE Access*, vol. 12, pp. 87407-87419, 2024. doi: 10.1109/ACCESS.2024.3409273.
- [5] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009. doi: 10.1109/CISDA.2009.5356528.
- [6] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," in *Proc. 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet'20)*, 2020. doi: 10.1145/3416013.3426457.
- [7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proc. 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108-116. doi: 10.5220/0006639801080116.
- [8] N. Koroniotis, N. Moustafa, E. Sitnikov, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779-796, 2019. doi: 10.1016/j.future.2019.05.041.
- [9] 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, 102994, 2021. doi: 10.1016/j.scs.2021.102994.
- [10] R. Alexander et al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *RFC 6550*, 2012. doi: 10.17487/RFC6550.
- [11] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," *RFC 7252*, 2014. doi: 10.17487/RFC7252.
- [12] L. Lin, Q. Zhong, J. Qiu, and Z. Liang, "E-GRACL: an IoT intrusion detection system based on graph neural networks," *The Journal of Supercomputing*, 2024/2025. doi: 10.1007/s11227-024-06471-5.
- [13] H. C. Altunay and Z. Albayrak, "A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks," *Journal of Engineering Science and Technology (JESTCH)*, 2023. doi: 10.1016/j.jestch.2022.101322.