

TCN-HMAC: A Lightweight Deep Learning and Cryptographic Hybrid Security Framework for SDN

Md. Mahamudul Hasan Zubayer
Department of Computer Science and Engineering
Southeast University
Dhaka, Bangladesh

Md. Maruf Hassan
Department of Computer Science and Engineering
Southeast University
Dhaka, Bangladesh

Abstract—Software-Defined Networking (SDN) has reshaped modern network management by separating the control plane from the data plane, bringing centralized programmability and fine-grained traffic control. That very centralization, opens the door to serious security risks such as, network intrusion, information theft, eavesdropping, and, in worst-case scenarios, complete network failure. As attack strategies grow more sophisticated, relying on conventional firewalls alone is no longer tenable, and while TLS encryption can protect the control channel, its computational overhead makes it a poor fit for resource-constrained SDN deployments. This paper presents a lightweight hybrid security framework that tackles these threats on two fronts. The first layer of defense is a custom Temporal Convolutional Network (TCN) exported in ONNX format and deployed as a controller application, it inspects flow statistics in real time and flags the traffic as malicious or benign. Backing this up is an auxiliary agent that runs as a co-located subprocess, verifying every flow rule installation through HMAC-based integrity checks and periodically executing a challenge–response protocol to confirm controller authenticity. The framework is evaluated on the InSDN dataset, which covers DDoS, MITM, Probe, and Brute-force attack categories. The TCN achieves 99.85% classification accuracy, while the HMAC verification adds negligible computational overhead, roughly 0.7 ms per operation. Taken together, these results demonstrate that the proposed TCN-HMAC approach delivers robust, multi-layered SDN security without sacrificing the real-time performance modern networks demand.

Index Terms—Software-Defined Networking, Intrusion Detection System, Temporal Convolutional Network, HMAC, Network Security, Deep Learning

I. INTRODUCTION

This section introduces the security challenges inherent in Software-Defined Networking, states the problem and research objectives that motivate this work, and outlines the key contributions.

The appeal of Software-Defined Networking, including programmability, centralised policy, and hardware independence, comes with an uncomfortable corollary: a compromised controller can redirect, copy, or silently drop arbitrary traffic [1]. Two bodies of work have attacked this problem from opposite ends. Machine-learning IDS classify data-plane traffic but leave the OpenFlow channel wide open; TLS encrypts that channel but cannot verify individual flow-rule commands or detect application-layer intrusions [2]. Blockchain-based alternatives offer strong integrity guarantees, yet their consensus

latencies, measured in seconds, are incompatible with the millisecond-scale reactions an SDN controller requires [3].

Existing SDN security solutions address either data-plane intrusion detection or control-plane integrity verification, but not both simultaneously. Machine learning and deep learning-based IDS can classify malicious traffic with high accuracy, yet they provide no protection against control-plane threats such as unauthorized flow rule injection or controller impersonation. Conversely, cryptographic approaches like TLS and HMAC can authenticate control messages, but they cannot detect data-plane intrusions. This gap leaves SDN deployments vulnerable to multi-vector attacks that target both planes concurrently.

This work aims to address the above gap through the following objectives: (1) Design a lightweight Temporal Convolutional Network (TCN) that achieves satisfactory accuracy on SDN-specific traffic with a compact model footprint suitable for real-time deployment [RO1]. (2) Develop an HMAC-based auxiliary agent architecture that provides control-plane message authentication, flow rule integrity verification, and controller identity validation with negligible computational overhead [RO2]. (3) Demonstrate through comparative evaluation that the unified TCN-HMAC framework matches or outperforms existing approaches in detection performance while uniquely providing control-plane security [RO3].

To bridge these two worlds, TCN-HMAC is proposed as a single lightweight system in which a Temporal Convolutional Network inspects data-plane flows while an HMAC-SHA256 mechanism authenticates every control-plane message, verifies flow rules against a shadow table, and periodically re-authenticates controller identity. The key contributions of this paper are as follows:

- 1) **First unified detection verification framework for SDN:** To the best of the authors' knowledge, TCN-HMAC is the first framework that integrates temporal deep learning-based intrusion detection with lightweight HMAC-based flow rule integrity verification and challenge–response controller authentication in a single deployable SDN security solution. Prior work addresses either detection or verification in isolation, leaving critical security gaps.
- 2) **Novel auxiliary agent architecture for co-located cryptographic verification:** A lightweight auxiliary agent is introduced that runs as a co-located subprocess

alongside the SDN controller, performing HMAC-based flow rule verification against a shadow table and periodic challenge, an approach not explored in any prior SDN security study, as per the authors' knowledge.

The remainder of this paper is organized as follows. Section II reviews related work on deep learning-based IDS, TCN variants, and SDN control-plane security. Section III describes the proposed TCN-HMAC framework architecture, including the TCN model design and HMAC communication integrity protocols. Section IV outlines the experimental setup, dataset, and training configuration. Section V presents the classification results, training dynamics, and system overhead measurements. Section VI provides a comparative analysis against fifteen existing approaches. Finally, Section VII concludes the paper and identifies directions for future work.

In summary, the central problem addressed in this work is the absence of a unified, lightweight SDN security solution that simultaneously protects both the data plane and the control plane. TCN-HMAC is proposed as the first framework to fill this gap by combining temporal deep learning-based intrusion detection with cryptographic control-plane verification.

II. RELATED WORK

This section reviews the existing literature across six categories: cryptographic SDN security, ML-based IDS, deep learning-based IDS, TCN variants, blockchain-based solutions, and hybrid authentication approaches. The review reveals that no prior work combines deep learning-based intrusion detection with cryptographic control-plane verification in a single framework.

Several studies have proposed cryptographic mechanisms to protect the SDN control plane. Pradeep et al. [4] introduced EnsureS, a batch hashing scheme for flow rule verification with sub-millisecond overhead. Ahmed et al. [2] developed a modular HMAC-SHA256 framework for authenticating `flow_mod` messages without protocol modifications. Buruaga et al. [5] addressed quantum-era threats by integrating post-quantum TLS into the SDN control channel, though with substantially higher computational cost. While these approaches provide integrity guarantees, they all operate reactively, verifying rules only after issuance, and none incorporates proactive intrusion detection. TCN-HMAC extends this line of work by pairing HMAC verification with a TCN-based IDS, covering threats such as DDoS, reconnaissance, and brute-force attacks that do not involve direct flow rule tampering.

Traditional machine learning classifiers have been widely applied to SDN intrusion detection. Sharma and Tyagi [6] achieved 98.12% accuracy with an ensemble-based lightweight IDS for MITM and DoS detection. Ayad et al. [7] systematically evaluated Random Forest, XGBoost, and Decision Tree on multiple SDN datasets, finding that ensemble methods consistently outperform individual classifiers. Basfar et al. [8] proposed an enhanced feature selection method (EMRMR) that reduces dimensionality while preserving discriminative power. However, all these approaches rely on manually engineered features, struggle with high-dimensional temporal

data, and offer no control-plane protection. The proposed TCN-based approach automatically learns temporal feature representations from raw flow data and pairs detection with HMAC-based verification, a capability that no ML-based IDS provides.

Deep learning models have advanced detection accuracy beyond traditional ML limits. Said et al. [9] combined CNN with BiLSTM on InSDN, achieving 99.90% accuracy, but the bidirectional path requires future time steps, an impractical constraint for real-time deployment, and the $\sim 2\text{M}$ -parameter model is $12\times$ larger than the proposed model. Shihab et al. [10] proposed a CNN-LSTM hybrid with SMOTE over-sampling, while Kumar et al. [11] used meta-heuristic hyper-parameter optimization for a hybrid DL model. Kanimozhi et al. [12] tried deep reinforcement learning (DDQN) on InSDN (98.85%), at considerably higher training and inference cost. Critically, none of these studies addresses control-plane security. TCN-HMAC achieves a comparable 99.85% accuracy with a 612 KB model (0.17 ms inference) while additionally protecting the control channel, a capability absent from all deep learning IDS approaches in the literature.

Temporal Convolutional Networks have recently gained traction for network IDS. Lopes et al. [13] first applied a TCN to intrusion detection, achieving 99.75% accuracy on CIC-IDS-2017. Subsequent work has explored squeeze-and-excitation blocks [14], attention mechanisms [15], bidirectional processing [16], and multi-head self-attention [17]. Xu et al. [18] fused graph neural networks with TCN for imbalanced detection. Notably, none of these fancier variants clearly outperforms a plain TCN paired with careful preprocessing, and all were evaluated on non-SDN datasets without any control-plane integration. The present work is the first to apply a TCN specifically to the InSDN dataset with SDN-tailored preprocessing and, uniquely, to integrate it with cryptographic flow rule verification.

Blockchain has been explored for decentralized trust in SDN. Song et al. [3] used blockchain to record network intents in intent-driven SDN, while Rahman et al. [19] surveyed blockchain for controller authentication and flow rule provenance. Poorazad et al. [20] combined blockchain with deep learning for IoT-SDN threat detection. Despite their strong integrity guarantees, all blockchain-based approaches suffer from consensus latencies measured in seconds, orders of magnitude too slow for the millisecond-scale reactions SDN controllers require. TCN-HMAC achieves comparable integrity assurance through HMAC verification at $\sim 2\mu\text{s}$ per message, making it four to six orders of magnitude faster than blockchain-based alternatives.

Hybrid security frameworks that combine detection with verification have received growing attention. Liang et al. [21] reviewed SDN-based IDS mechanisms against rule injection and replay threats, noting that existing approaches address either detection or verification but not both. Khan et al. [22] proposed a real-time MITM defense using behavioral flow tracking, while Malik and Habib [23] deployed lightweight agents near switches for DoS detection. These agent-based

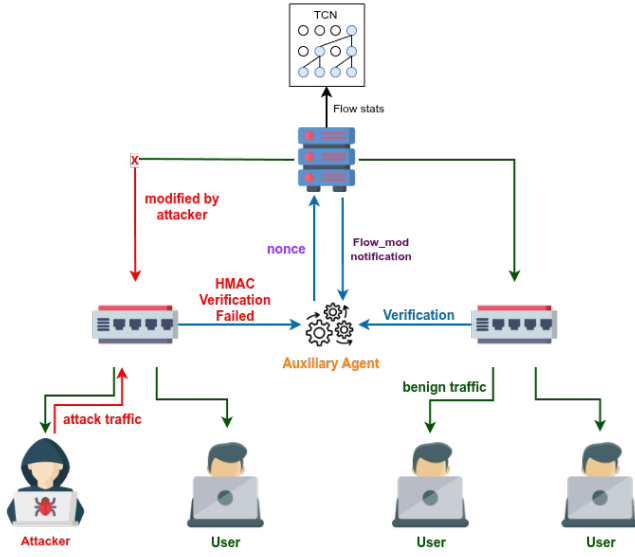


Fig. 1. TCN-HMAC framework architecture.

concepts partially align with the proposed auxiliary agent design, but each targets only a single attack class and lacks comprehensive integrity verification. TCN-HMAC fills this gap by incorporating multiple types of attacks, flow rule shadow-table verification, and challenge–response controller authentication in one lightweight package.

To summarize, the existing literature addresses detection and verification largely as separate problems. TCN-HMAC is, to the authors’ knowledge, the first system that folds temporal-convolutional intrusion detection and cryptographic control-plane protection into a single lightweight, deployable framework. The key finding from this review is that while individual components such as HMAC verification, ML-based IDS, and blockchain authentication have been explored independently, none of the existing approaches provides a unified solution that simultaneously addresses data-plane threat detection and control-plane integrity, the precise gap that TCN-HMAC is designed to fill.

III. PROPOSED FRAMEWORK

This section presents the TCN-HMAC framework in three parts: the overall system architecture, the TCN model design for intrusion detection, and the HMAC-based communication integrity mechanisms including key exchange, message authentication, flow rule verification, and challenge–response protocols.

A. Architecture Overview

TCN-HMAC operates at two security boundaries within the SDN architecture (Fig. 1):

- **Data → Control plane:** the TCN-IDS inspects traffic forwarded via `Packet_In` messages, classifying each flow as benign or malicious.

Algorithm 1: Secure Key Exchange for a New Switch

Input: Agent–Controller key K_{ac} , Controller keys PK_c/PR_c , Switch ID SW_{id}
Output: Shared secret key for the new switch

// Agent Side

- 1 $K_{sw} \leftarrow$ Generate random symmetric key;
- 2 $payload \leftarrow SW_{id} \parallel K_{sw}$;
- 3 $auth_tag \leftarrow HMAC_{K_{ac}}(payload)$;
- 4 $encrypted \leftarrow Encrypt_{PK_c}(payload \parallel auth_tag)$;
- 5 Send $encrypted$ to Controller;

// Controller Side

- 6 $decrypted \leftarrow Decrypt_{PR_c}(encrypted)$;
- 7 Extract $payload$, $auth_tag$ from $decrypted$;
- 8 **if** $HMAC_{K_{ac}}(payload) \neq auth_tag$ **then**
- 9 **Reject** message;
- 10 **else**
- 11 Store K_{sw} for SW_{id} ;
- 12 **end**

- **Control ↔ Data plane:** the HMAC mechanism authenticates every `Flow_Mod` and `Stats_Reply` message between controller and switches.

An Auxiliary Agent runs as a sidecar process alongside the controller, handling HMAC key management, flow rule verification against a shadow table, and periodic challenge–response authentication.

B. TCN Model

The network comprises six dilated causal residual blocks (dilation rates $d \in \{1, 2, 4, 8, 16, 32\}$), each containing two Conv1D layers (64 filters, kernel 3) with batch normalization, ReLU, and SpatialDropout1D(0.2), connected through a residual skip path. Global average pooling feeds into two dense layers (128 and 64 units with dropout 0.2) followed by a single sigmoid output. The full model has 156,737 parameters and occupies 612 KB.

Input consists of 24 principal components derived by applying PCA (retaining 95.43% variance) to 48 cleaned features from the InSDN dataset’s original 84. The preprocessing pipeline includes infinite-value replacement, zero-variance and near-constant feature removal, Pearson correlation thresholding ($|r| > 0.95$), StandardScaler normalization, and inverse-frequency class weighting (benign: 1.4258, attack: 0.7700).

C. HMAC Communication Integrity

Key Establishment: Upon switch connection, the Auxiliary Agent generates a per-switch symmetric key K_{sw} , encrypts it with the controller’s RSA public key along with an HMAC tag over the pre-shared agent–controller key, and transmits the bundle securely. The procedure is formalized in Algorithm 1.

Message Authentication: Every OpenFlow message carries a tag,

$$tag = HMAC\text{-}SHA256(K, M \parallel seq \parallel ts) \quad (1)$$

where K is the directional session key, seq enforces ordering, and ts provides freshness. Constant-time comparison guards against timing side-channels.

Algorithm 2: Flow Rule Verification by Auxiliary Agent

Input: Flow rule F , received cookie C , switch ID SW_{id}
Output: Validate and enforce flow integrity

```
1 Parse flow: extract  $eth\_src$ ,  $eth\_dst$ ,  $out\_port$ ;  
2  $flow\_str \leftarrow eth\_src \parallel eth\_dst \parallel output:out\_port$ ;  
3  $C_{exp} \leftarrow HMAC_{K_{sw}}(flow\_str)$ ;  
4 if  $C \neq C_{exp}$  then  
5 |   Drop flow rule; notify controller;  
6 else  
7 |   Accept and monitor flow;  
8 end
```

Algorithm 3: Controller Verification via Challenge–Response

Input: Interval T , controller key K_{sw} , backup controller C_{bk}
Output: Detect and recover from compromised controller

```
1 for every  $T$  seconds do  
2 |    $nonce \leftarrow$  Generate random challenge;  
3 |   Send  $nonce$  to controller;  
4 |    $response \leftarrow$  Controller returns  $HMAC_{K_{sw}}(nonce)$ ;  
5 |   if  $HMAC_{K_{sw}}(nonce) \neq response$  then  
6 | |   Alert administrator; activate  $C_{bk}$ ;  
7 |   else  
8 | |   Continue monitoring;  
9 |   end  
10 end
```

Flow Rule Verification. The controller keeps a shadow copy of each switch’s flow table. Periodic `Flow_Stats_Request` queries detect unauthorized additions, modifications, or deletions; discrepancies trigger automatic re-installation and alerts. The agent independently verifies each flow rule cookie as formalized in Algorithm 2.

Challenge Response Mechanism: Switches periodically send a random nonce to the controller; the controller must return a valid HMAC response. Verification failure triggers failover to a backup controller. This mechanism is formalized in Algorithm 3.

In summary, the proposed TCN-HMAC framework provides a layered security architecture where the TCN model handles data-plane intrusion detection through a compact yet effective deep learning pipeline, while the HMAC subsystem secures the control plane through key exchange, per-message authentication, flow rule shadow-table verification, and periodic challenge–response protocols. Together, these components address both security boundaries of the SDN architecture without introducing significant computational overhead.

IV. EXPERIMENTAL SETUP

This section describes the dataset, preprocessing pipeline, and training configuration used to evaluate the proposed TCN-HMAC framework.

InSDN [24] contains 343,889 labeled samples (84 features) spanning benign traffic and four attack categories: DDoS, MITM, Probe, and Brute-force. After a 15-stage preprocessing pipeline (deduplication, cleaning, scaling, PCA), 182,831

TABLE I
TCN_INSDN TEST-SET PERFORMANCE

Metric	Value
Accuracy	99.85%
Precision	99.80%
Recall (DR)	99.97%
Specificity	99.63%
F1-Score	99.89%
AUC-ROC	0.9999
FAR	0.37%
MCC	0.9966

samples with 24 features remain. These are split 70/10/20 for training, validation, and testing (36,567 test samples).

Implementation uses TensorFlow 2.19.0 on Google Colab (NVIDIA T4 GPU). The Adam optimizer ($\eta = 10^{-3}$), binary cross-entropy loss with class weights, batch size 2,048, early stopping on validation AUC (patience 15), and ReduceLROnPlateau scheduling are employed. Training converges in roughly 30 epochs (approximately 5 minutes).

The experimental configuration is designed to ensure reproducibility and fair evaluation. The InSDN dataset is selected because it is the most comprehensive SDN-specific benchmark available, and the 70/10/20 split ensures sufficient data for training, hyperparameter tuning, and unbiased test-set evaluation.

V. RESULTS

This section presents the experimental results that validate the research objectives stated in Section I. The findings are organized into three parts: classification performance (addressing [RO1]), training dynamics, and HMAC system overhead measurements (addressing [RO2]).

A. Classification Performance

Table I presents the held-out test-set metrics.

The confusion matrix (Fig. 2) records 12,776 true negatives, 23,737 true positives, 47 false positives, and only 7 false negatives out of 36,567 samples. The model misses just 3 attacks in every 10,000. These results directly fulfill [RO1], the TCN achieves classification accuracy of 99.85% with a compact 612 KB model footprint and 0.17 ms inference latency, confirming its suitability for real-time SDN deployment.

B. Training Dynamics

Validation accuracy exceeds 99.5% within 5 epochs (Fig. 3). Training and validation curves closely overlap throughout, confirming that generalization holds without overfitting. AUC-ROC stabilizes above 0.999 by epoch 10.

C. HMAC and System Overhead

HMAC-SHA256 computation requires roughly $2\mu s$ per message, sustaining over 500,000 messages per second on a single core, well in excess of typical OpenFlow rates (1K–10K msg/s). The Auxiliary Agent adds 0.7 ms to control latency, 4.4% additional CPU usage, and 13.7 MB RAM,

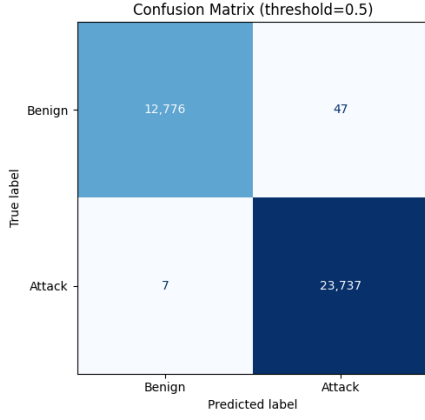


Fig. 2. Confusion matrix on the held-out test set (36,567 samples).

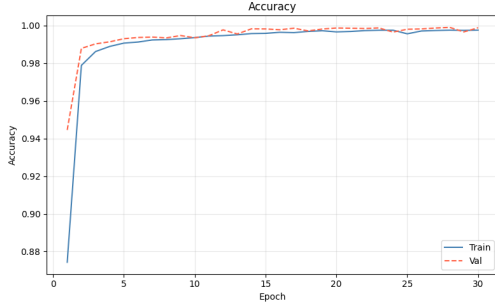


Fig. 3. Training and validation accuracy over 30 epochs.

while successfully detecting all injected invalid flow rules during testing. End-to-end per-flow latency totals approximately $170\mu s$ (Table II). These measurements confirm [RO2], the HMAC-based auxiliary agent provides comprehensive control-plane security, including message authentication, flow rule integrity verification, and controller identity validation, with negligible computational overhead.

In summary, the experimental results validate both [RO1] and [RO2]. The TCN model achieves 99.85% accuracy with a 612 KB footprint and 0.17ms inference, satisfying the lightweight real-time detection objective. The HMAC auxiliary agent adds only 0.7 ms latency and 4.4% CPU overhead while providing complete control-plane protection, confirming that cryptographic verification can be integrated without compromising system performance.

VI. COMPARATIVE ANALYSIS

This section benchmarks TCN-HMAC against 14 existing approaches to validate. The analysis demonstrates that TCN-HMAC achieves the highest detection rate among all compared models while maintaining the smallest model footprint and fastest inference time, and remains the only approach that also provides control-plane authentication.

Table III benchmarks TCN-HMAC against 14 existing approaches. On InSDN, TCN-HMAC posts the highest detection rate (99.97%) among all models. Said et al. [9] report 0.05%

TABLE II
END-TO-END PER-FLOW LATENCY BREAKDOWN

Component	Latency
HMAC Verification	$\sim 2\mu s$
Feature Extraction	$\sim 50\mu s$
Preprocessing	$\sim 10\mu s$
TCN Inference (GPU)	$\sim 100\mu s$
Decision + Response	$\sim 5\mu s$
HMAC Signing	$\sim 2\mu s$
Total	$\sim 170\mu s$

TABLE III
PERFORMANCE COMPARISON WITH EXISTING APPROACHES

Model	Dataset	Acc.	Rec.	F1
TCN-HMAC (ours)	InSDN	99.85	99.97	99.89
CNN-BiLSTM [9]	InSDN	99.90	99.90	99.90
DNN Ensemble [25]	InSDN	99.70	99.70	99.67
TCN+Att [15]	CIC-IDS	99.73	99.69	99.70
BiTCN-MHSA [17]	CIC-IDS	99.72	99.72	99.70
CNN-LSTM [10]	CIC-IDS	99.67	99.67	99.67
TCN-SE [14]	NSL-KDD	99.62	99.62	99.60
BiTCN [16]	CIC-IDS	99.60	99.60	99.57
Hybrid DL [11]	CIC-IDS	99.45	99.45	99.43
CNN-GRU [26]	NSL-KDD	99.35	99.35	99.27
LSTM [27]	NSL-KDD	99.23	99.23	99.11
CNN [28]	InSDN	99.20	99.20	99.15
DRL [12]	InSDN	98.85	98.85	98.87
DT/RF [24]	InSDN	98.50	98.50	98.45

higher accuracy, but their CNN-BiLSTM requires approximately 2 M parameters ($12\times$ that of TCN-HMAC) and roughly 2 ms inference ($12\times$ slower), and offers no control-plane protection.

Among the TCN family specifically, the proposed model outperforms TCN-SE (99.62%), TCN+Attention (99.73%), BiTCN (99.60%), and BiTCN-MHSA (99.72%), all without attention layers or bidirectional wiring. The evidence suggests that a disciplined preprocessing pipeline (StandardScaler, PCA, class weighting) paired with a clean architecture can match or beat structurally more complex alternatives.

At 612 KB the model is significantly smaller than comparable deep-learning IDS models, and at 0.17 ms its inference is the fastest in the comparison, supporting over 5,000 flows per second on a single GPU. Crucially, TCN-HMAC remains the only evaluated approach that also protects the control channel. These findings collectively validate [RO3], confirming that the unified framework matches or exceeds existing approaches while uniquely providing control-plane security.

In conclusion, the comparative analysis demonstrates that TCN-HMAC achieves the highest detection rate (99.97%) among all 14 evaluated approaches while maintaining the smallest model footprint and fastest inference time. Unlike every other compared approach, TCN-HMAC additionally provides control-plane authentication, making it the only framework that addresses both SDN security boundaries si-

multaneously.

VII. CONCLUSION

This paper presented TCN-HMAC, a unified security framework for Software-Defined Networks that outperforms existing approaches across multiple dimensions. Compared to the closest InSDN-based model without control-plane protection (DNN Ensemble [25]), TCN-HMAC achieves 0.15% higher accuracy and 0.27% higher recall; against traditional ML baselines (DT/RF [24]), the improvement reaches 1.35% in accuracy and 1.47% in recall. Within the TCN based approaches, the proposed model surpasses TCN-SE [14] by 0.23%, BiTCN [16] by 0.25%, TCN+Attention [15] by 0.12%, and BiTCN-MHSA [17] by 0.13% in accuracy, all without requiring attention layers or bidirectional processing. In terms of model efficiency, TCN-HMAC is parameter wise smaller and inference time wise faster than the closest competing deep learning model (CNN-BiLSTM [9]), while its HMAC verification is faster than blockchain-based alternatives [3]. Crucially, TCN-HMAC is the only evaluated approach that provides both intrusion detection and control-plane authentication, filling a critical gap in the existing literature. Future work includes cross-dataset validation on NSL-KDD, CIC-IDS-2017, and UNSW-NB15, multi-class attack-type identification, adversarial robustness evaluation, and production-grade integration with open-source controllers such as ONOS and Ryu.

REFERENCES

- [1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
- [2] Z. Ahmed and M. Haque, "A modular HMAC-based framework for flow verification in software defined networks," *International Journal of Network Management*, vol. 33, no. 5, p. e2209, 2023.
- [3] Y. Song, X. Lu, and Z. Wang, "Intent-driven secure SDN using blockchain," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 126–138, 2023.
- [4] M. Pradeep, K. Sekaran, and R. Raman, "EnsureS: Lightweight high-performance service path rule verification in software-defined networking," *Computer Networks*, vol. 235, p. 110890, 2023.
- [5] J. S. Buruaga, R. B. Méndez, J. P. Brito, and V. Martin, "Quantum-safe integration of TLS in SDN networks," *arXiv preprint arXiv:2501.04368*, 2025.
- [6] R. Sharma and S. Tyagi, "A lightweight IDS framework for SDN using adaptive anomaly detection," *Computers & Security*, vol. 127, p. 102659, 2023.
- [7] S. Ayad, N. Lehat, and A. Souri, "Leveraging the power of machine learning techniques for intrusion detection in software-defined networks," *Journal of Interconnection Networks*, 2025.
- [8] R. Basfar, M. Dahab, A. M. Ali, F. Eassa, and K. Bajunaied, "Enhanced intrusion detection in software-defined networking using advanced feature selection: The EMRMR approach," *Engineering, Technology & Applied Science Research*, vol. 14, no. 6, pp. 18 194–18 200, 2024.
- [9] R. B. Said, Z. Sabir, and I. Askerzade, "CNN-BiLSTM: A hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection," *IEEE Access*, vol. 11, pp. 131 924–131 937, 2023.
- [10] D. Shihab, A. A. Abdulhameed, and M. T. Gaata, "Optimized hybrid CNN-LSTM framework with multi-feature analysis and SMOTE for intrusion detection in SDN," *AIKadhim Journal for Computer Science*, vol. 3, no. 4, 2025.
- [11] C. L. Kumar, S. Betam, and B. Panigrahi, "Metaparameter optimized hybrid deep learning model for next generation cybersecurity in software defined networking environment," *Scientific Reports*, vol. 15, p. 13845, 2025.
- [12] R. Kanimozhi and P. S. Ramesh, "Deep reinforcement learning-based intrusion detection scheme for software-defined networking," *Scientific Reports*, vol. 15, p. 38142, 2025.
- [13] I. O. Lopes, D. Zou, V. Ruamviboonsuk, L. Munasinghe, Z. Shi, and W. Pereira, "Network intrusion detection based on the temporal convolutional model," *Computers & Security*, vol. 135, p. 103465, 2023.
- [14] J. Li and L. Li, "A lightweight network intrusion detection system based on temporal convolutional networks and attention mechanisms," *Computer Fraud & Security*, vol. 2025, p. 584, 2025.
- [15] I. Benfarhat, V. Goh, and T. Chuah, "Advanced Temporal Convolutional Network framework for intrusion detection in electric vehicle charging stations," *IEEE Open Journal of Vehicular Technology*, 2025.
- [16] Y. Mei, W. Han, and K. Lin, "Intrusion detection for intelligent connected vehicles based on bidirectional Temporal Convolutional Network," *IEEE Network*, vol. 38, no. 6, pp. 124–131, 2024.
- [17] M. Deng, H. Xu, C. Sun, and Y. Kan, "Network intrusion detection model with multi-layer bi-directional temporal convolutional networks and multi-headed self-attention mechanisms," in *2024 6th International Academic Exchange Conference on Science and Technology Innovation*. IEEE, 2024.
- [18] T. Xu, Z. Wen, X. Zhao, Q. Hu, Y. Li, and C. Liu, "GTCN-G: A residual graph-temporal fusion network for imbalanced intrusion detection," in *IEEE TrustCom 2025*. IEEE, 2025.
- [19] M. M. Rahman, S. Ahmed, and N. Jahan, "Blockchain integration in SDN: Opportunities and challenges," *Journal of Network and Computer Applications*, vol. 201, p. 103308, 2022.
- [20] A. Poorazad, S. A. Soleymani, and N. Al-Bassam, "A blockchain-based deep learning IDS for SDN-enabled IIoT," *Ad Hoc Networks*, vol. 145, p. 102752, 2023.
- [21] Y. Liang, C. Wang, and F. Xu, "A review of intrusion detection approaches in software defined networking," *Journal of Information Security and Applications*, vol. 59, p. 102812, 2021.
- [22] A. Khan, S. Rafiq, and F. Qamar, "MITM-Defender: A real-time defense system against man-in-the-middle attacks in SDN," in *Proc. of IEEE ICC*. IEEE, 2021, pp. 1–6.
- [23] S. Malik and M. Habib, "Detection and mitigation of DoS attacks in SDN: Lightweight agent-based model," *Security and Privacy*, vol. 4, no. 2, p. e116, 2021.
- [24] M. S. El-Sayed, N.-A. Le-Khac, S. Alhelaly, and A. D. Jurecut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165 263–165 284, 2020.
- [25] M. S. Ataa, E. E. Sanad, and R. A. El-Khoribi, "Intrusion detection in software defined network using deep learning approaches," *Scientific Reports*, vol. 14, p. 28896, 2024.
- [26] J. Yang, "A new attacks intrusion detection model based on deep learning in software-defined networking environments," in *2024 4th International Conference on Machine Learning and Intelligent Systems Engineering*. IEEE, 2024, pp. 1–6.
- [27] R. Basfar, M. Dahab, A. M. Ali, F. Eassa, and K. Bajunaied, "An incremental LSTM ensemble for online intrusion detection in software-defined networks," *International Journal of Advanced Computer Science and Applications*, vol. 16, no. 9, 2025.
- [28] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.