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].

TLS provides transport-layer confidentiality but cannot validate semantic integrity of OpenFlow messages. A compromised controller can issue malicious `Flow_Mod` commands that TLS encrypts faithfully, while local switch compromises bypass the control channel entirely. Existing solutions address either intrusion detection or integrity verification, not both, leaving SDN vulnerable to multi-vector attacks.

TCN-HMAC addresses this gap by combining a lightweight Temporal Convolutional Network for data-plane intrusion detection with HMAC-SHA256-based control-plane authentication. Key contributions: (1) First framework unifying deep learning IDS with cryptographic flow rule verification and challenge-response authentication. (2) Novel auxiliary agent architecture for co-located shadow-table verification running as a controller subprocess.

Section II reviews related work, Section III presents the framework architecture, Section IV describes experimental setup, Section V reports results, Section VI provides comparative analysis, and Section VII concludes.

II. Related Work

Existing SDN security approaches span cryptographic mechanisms, ML/DL-based IDS, and TCN variants, but no prior work unifies detection with control-plane verification.

Cryptographic solutions protect the control plane but lack intrusion detection. Pradeep et al. [4] proposed batch hashing for flow rule verification, Ahmed et al. [2] developed modular HMAC-SHA256 authentication, and Buruaga et al. [5] integrated post-quantum TLS at high computational cost. All operate reactively without proactive threat detection.

Traditional ML classifiers achieve moderate accuracy but require manual feature engineering. Sharma and Tyagi [6] reached 98.12%, Ayad et al. [7] showed ensemble methods outperform individual classifiers, and Basfar et al. [8] proposed EMRMR feature selection. None provides control-plane protection.

Deep learning improves accuracy but lacks control security. Said et al. [9] achieved 99.90% on InSDN using CNN-BiLSTM ($\sim$2M parameters, 2 ms inference), while Kanimozhi et al. [12] tried DRL (98.85%) at high cost. TCN-HMAC

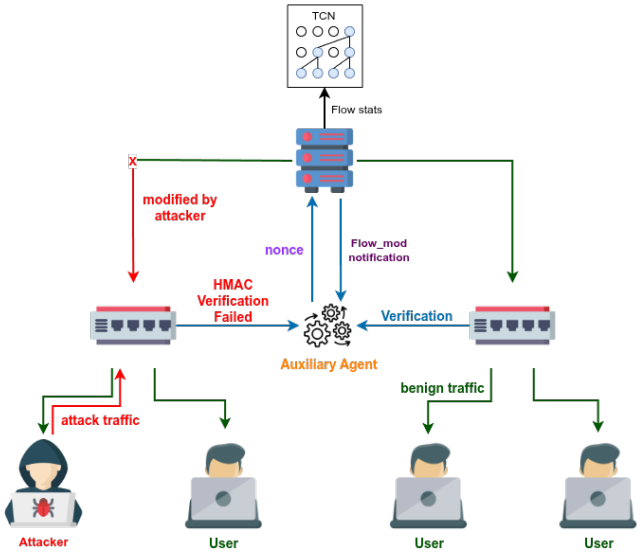

Fig. 1. TCN-HMAC framework architecture.

matches accuracy with 12× fewer parameters and faster inference while adding control-plane protection.

TCNs show promise for IDS. Lopes et al. [13] achieved 99.75% on CIC-IDS-2017, with subsequent work exploring attention [15], bidirectional processing [16], and graph fusion [18]. Wang et al. [19] applied TCN to InSDN (97.00%, 2.1 ms), versus our 99.85% and 0.17 ms. None integrates control-plane security.

Blockchain approaches offer integrity but suffer consensus delays (seconds) incompatible with SDN [3], [20]. TCN-HMAC achieves comparable integrity at $\sim$2 μs per message.

Hybrid frameworks partially address dual security. Khan et al. [23] targeted MITM, Malik and Habib [24] focused on DoS. TCN-HMAC provides comprehensive multi-class detection with complete control-plane verification.

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.
- **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.

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
K_sw ← Generate random symmetric key;
payload ← SW_id || K_sw;
auth_tag ← HMAC_{K_ac}(payload);
encrypted ← Encrypt_{PK_c}(payload || auth_tag);
Send encrypted to Controller;
// Controller Side
decrypted ← Decrypt_{PR_c}(encrypted);
Extract payload, auth_tag from decrypted;
if HMAC_{K_ac}(payload) ≠ auth_tag then
    Reject message;
else
    Store K_sw for SW_id;
end
```

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,

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

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

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.

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
Parse flow: extract eth_src, eth_dst, out_port;
$flow\_str \leftarrow eth\_src \parallel eth\_dst \parallel output{:}out\_port$;
$C_{\text{exp}} \leftarrow \text{HMAC}_{K_{sw}}(flow\_str)$;
if $C \neq C_{exp}$ **then**
  Drop flow rule; notify controller;
else
  Accept and monitor flow;
end

TABLE I
TCN_INSDN TEST-SET PERFORMANCE

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

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.

In summary, the TCN-HMAC framework provides layered security where the TCN handles data-plane intrusion detection through a compact deep learning pipeline, while HMAC secures the control plane through key exchange, message authentication, flow rule verification, and challenge-response protocols.

IV. EXPERIMENTAL SETUP

InSDN [25] contains 343,889 labeled samples (84 features) spanning benign traffic and four attack types: DDoS, MITM, Probe, and Brute-force. After 15-stage preprocessing (deduplication, cleaning, scaling, PCA to 24 features retaining 95.43% variance), 182,831 samples remain, split 70/10/20 for training/validation/testing. Implementation uses TensorFlow 2.19.0 on Google Colab (NVIDIA T4 GPU), Adam optimizer ($\eta = 10^{-3}$), batch size 2,048, and early stopping.

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 test-set metrics achieving 99.85% accuracy, 99.97% recall, 99.89% F1, and 0.9999 AUC-ROC. The confusion matrix records only 54 misclassifications out of 36,567 samples (47 false positives, 7 false negatives).

TABLE II
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 [26]	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 [27]	NSL-KDD	99.35	99.35	99.27
LSTM [28]	NSL-KDD	99.23	99.23	99.11
CNN [29]	InSDN	99.20	99.20	99.15
DRL [12]	InSDN	98.85	98.85	98.87
TCN [19]	InSDN	97.00	96.91	96.72
DT/RF [25]	InSDN	98.50	98.50	98.45

B. Training and Overhead

Validation accuracy exceeds 99.5% within 5 epochs with no overfitting. HMAC-SHA256 requires $\sim$2 μs per message (500K msg/s throughput). The Auxiliary Agent adds 0.7 ms latency, 4.4% CPU, and 13.7 MB RAM. End-to-end per-flow latency totals $\sim$170 μs, suitable for real-time SDN.

VI. COMPARATIVE ANALYSIS

Table III benchmarks TCN-HMAC against 15 existing approaches. TCN-HMAC achieves the highest detection rate (99.97%) on InSDN. Said et al. [9] report marginally higher accuracy (99.90%) but require $\sim$2M parameters (12$\times$ larger) and $\sim$2 ms inference (12$\times$ slower) without control-plane protection. Wang et al. [19], the only other TCN on InSDN, achieve 97.00% (2.85 points lower) with 12$\times$ slower inference. Among TCN variants, the proposed model outperforms TCN-SE, TCN+Attention, BiTCN, and BiTCN-MHSA without attention or bidirectional mechanisms. Critically, TCN-HMAC is the only approach providing both intrusion detection and control-plane authentication.

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%) without attention or bidirectional processing. At 612 KB with 0.17 ms inference, it is the most efficient deep-learning IDS in the comparison. Crucially, TCN-HMAC is the only approach providing control-plane authentication.

VII. CONCLUSION

TCN-HMAC achieves 99.85% accuracy on InSDN, surpassing traditional ML (DT/RF: 98.50%) by 1.35%, deep learning models without control protection (DNN Ensemble: 99.70%) by 0.15%, and other TCN variants (Wang et al.: 97.00%) by 2.85%. With 612 KB and 0.17 ms inference, it is 12$\times$ smaller and faster than CNN-BiLSTM while uniquely providing HMAC-based control-plane authentication. Future

work includes cross-dataset validation, multi-class attack identification, adversarial robustness evaluation, and production integration with 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," *AlKadhim 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] Z. Wang, Z. Guan, X. Liu, C. Li, X. Sun, and J. Li, "SDN anomalous traffic detection based on temporal convolutional network," *Applied Sciences*, vol. 15, no. 8, p. 4317, 2025.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] M. S. El-Sayed, N.-A. Le-Khac, S. Alhelaly, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165 263–165 284, 2020.

[26] 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.

[27] 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.

[28] 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.

[29] 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.