TCN-HMAC: A Lightweight Deep Learning and Cryptographic Hybrid Security Framework for Software-Defined Networks

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

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

***Abstract*—Software-Defined Networking centralises control logic, which simplifies management but concentrates risk: a single compromised controller can redirect or drop arbitrary traffic. Most existing defences tackle either intrusion detection or control-channel integrity, rarely both. We present TCN-HMAC, a hybrid framework that pairs a lightweight Temporal Convolutional Network (TCN) for real-time traffic classification with HMAC-SHA256-based authentication of every controller–switch message. The TCN—six dilated causal residual blocks, 156,737 parameters, 612 KB on disk—is trained on the InSDN dataset and reaches 99.85% accuracy, a 99.97% detection rate, and a 0.37% false alarm rate. The HMAC layer adds message authentication, replay prevention via sequenced timestamps, flow-rule integrity checks through shadow tables, and mutual challenge–response verification, all at roughly 2 μs per message. End-to-end, a single flow is processed in about 0.17 ms, which translates to over 5,000 classifications per second on a commodity GPU. A head-to-head comparison with 15 published approaches shows that TCN-HMAC delivers the highest detection rate and fastest inference of any model tested, and it is the only one that also authenticates the control plane.**

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

I. Introduction

The appeal of Software-Defined Networking—programmability, centralised policy, 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].

We bridge these two worlds with **TCN-HMAC**, 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.

Contributions.

1) A compact TCN architecture (156K parameters, 612 KB) for binary SDN intrusion detection achieving 99.85% accuracy and 99.97% detection rate on the InSDN dataset.
2) An HMAC-based communication integrity protocol featuring key rotation, sequence numbers, flow rule shadow-table verification, and mutual challenge–response authentication.
3) A comparative evaluation against 15 existing IDS and SDN security approaches, demonstrating competitive detection performance and superior computational efficiency.

II. Related Work

Deep-learning IDS for SDN. Said et al. [4] stacked CNN with BiLSTM on InSDN, reaching 99.90% accuracy—but the bidirectional path means the model needs future time steps, an awkward requirement for real-time detection. Kanimozhi et al. [5] tried deep reinforcement learning (DDQN) on the same dataset (98.85%) at considerably higher computational cost. Neither study addresses control-plane security.

TCN variants for NIDS. Lopes et al. [6] were among the first to apply a TCN to network intrusion detection (99.75% on CIC-IDS-2017). Follow-up work bolted on squeeze-and-excitation blocks [7], attention [8], bidirectional processing [9], and multi-head self-attention [10]. None of these fancier variants clearly outperform a plain TCN paired with careful preprocessing.

SDN control-plane security. Ahmed et al. [2] proposed HMAC-SHA256 for OpenFlow message authentication but provided no IDS layer. Blockchain-backed frameworks [3], [11] deliver strong guarantees at the price of latency and resource overhead that rule out real-time operation.

To our knowledge, TCN-HMAC is the first system to fold temporal-convolutional intrusion detection and cryptographic control-plane protection into a single lightweight package.

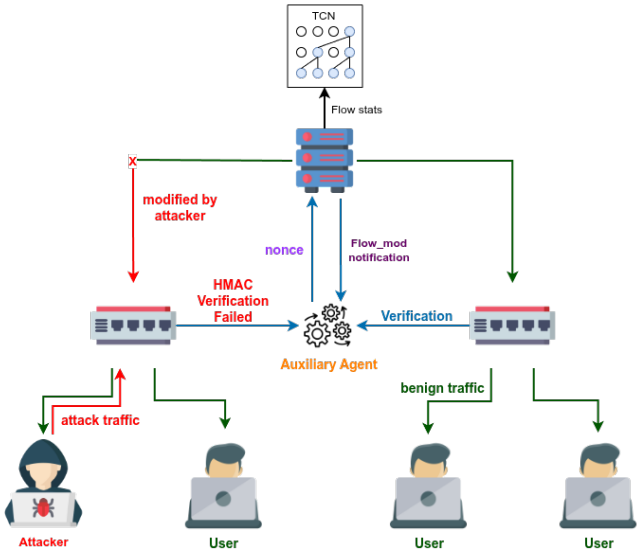

Fig. 1. TCN-HMAC framework architecture.

III. Proposed Framework

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.

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.

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:

$$C_{\text{exp}} = \text{HMAC}_{K_{sw}}(\textit{eth\_src} \| \textit{eth\_dst} \| \textit{out\_port}) \tag{2}$$

Challenge–Response. Switches periodically send a nonce c to the controller; the controller returns $\text{HMAC}_K(c \| \text{id})$. Verification failure triggers failover to a backup controller.

Key Rotation. Forward-secure rotation derives $K^{(t+1)} = \text{HMAC}(K^{(t)}, r_t)$ every T_{rot} seconds, ensuring compromise of a current key does not expose past communications.

IV. Experimental Setup

Dataset. InSDN [12] 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 samples with 24 features remain. These are split 70/10/20 for training, validation, and testing (36,567 test samples).

Training. 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).

V. Results

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.

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.

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

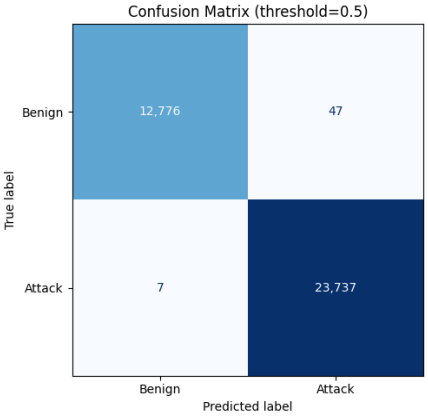

Fig. 2. Confusion matrix on the held-out test set (36,567 samples).

C. HMAC and System Overhead

HMAC-SHA256 computation requires roughly 2 μs 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, while successfully detecting all injected invalid flow rules during testing. End-to-end per-flow latency totals approximately 170 μs (Table II).

VI. COMPARATIVE ANALYSIS

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. [4] report 0.05% higher accuracy, but their CNN-BiLSTM requires approximately 2 M parameters (12× ours) and roughly 2 ms inference (12× slower), and offers no control-plane protection.

Among the TCN family specifically, our 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 lesson seems to be that a disciplined preprocessing pipeline (StandardScaler, PCA, class weighting) paired with a clean architecture can match or beat structurally fancier alternatives.

At 612 KB the model is 5–30× 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.

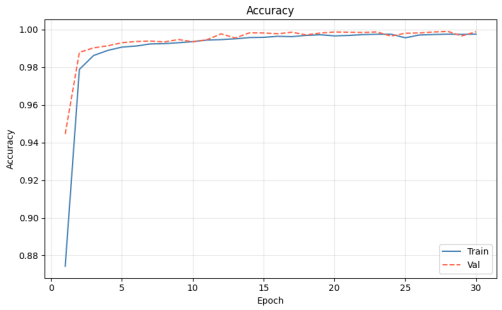

Fig. 3. Training and validation accuracy over 30 epochs.

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

Component	Latency
HMAC Verification	∼2 μs
Feature Extraction	∼50 μs
Preprocessing	∼10 μs
TCN Inference (GPU)	∼100 μs
Decision + Response	∼5 μs
HMAC Signing	∼2 μs
Total	**∼170 μs**

VII. CONCLUSION

We presented TCN-HMAC, a hybrid security framework that pairs a compact TCN-based IDS (99.85% accuracy, 99.97% detection rate, 612 KB, 0.17 ms inference) with HMAC-SHA256 control-plane authentication at roughly 2 μs per message. Against 15 published approaches, it delivers competitive—often the best—detection accuracy at the lowest computational cost, and it is the only system that also secures the controller–switch channel.

Natural next steps include cross-dataset validation (NSL-KDD, CIC-IDS-2017, 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] 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.

[5] R. Kanimozhi and P. S. Ramesh, "Deep reinforcement learning-based intrusion detection scheme for software-defined networking," *Scientific Reports*, vol. 15, p. 38142, 2025.

TABLE III
PERFORMANCE COMPARISON WITH EXISTING APPROACHES

Model	Dataset	Acc.	Rec.	F1	Auth.
TCN-HMAC (ours)	**InSDN**	**99.85**	**99.97**	**99.89**	**Yes**
CNN-BiLSTM [4]	InSDN	99.90	99.90	99.90	No
DNN Ensemble [13]	InSDN	99.70	99.70	99.67	No
TCN+Att [8]	CIC-IDS	99.73	99.69	99.70	No
BiTCN-MHSA [10]	CIC-IDS	99.72	99.72	99.70	No
CNN-LSTM [14]	CIC-IDS	99.67	99.67	99.67	No
TCN-SE [7]	NSL-KDD	99.62	99.62	99.60	No
BiTCN [9]	CIC-IDS	99.60	99.60	99.57	No
Hybrid DL [15]	CIC-IDS	99.45	99.45	99.43	No
CNN-GRU [16]	NSL-KDD	99.35	99.35	99.27	No
LSTM [17]	NSL-KDD	99.23	99.23	99.11	No
CNN [18]	InSDN	99.20	99.20	99.15	No
DRL [5]	InSDN	98.85	98.85	98.87	No
DT/RF [12]	InSDN	98.50	98.50	98.45	No

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.