RESEARCH-ARTICLE

A Trustworthy and Comprehensive Communication Auditing System for High-Performance Gateways

YU JIN, Nanjing University, Nanjing, Jiangsu, China

DAMING HUANG, Nanjing University, Nanjing, Jiangsu, China

JINCHENG XIANG, Nanjing University, Nanjing, Jiangsu, China

CHEN TIAN, Nanjing University, Nanjing, Jiangsu, China

Open Access Support provided by:

Nanjing University

Published: 01 December 2025

Citation in BibTeX format

CoNEXT '25:The 21st International Conference on emerging Networking EXperiments and Technologies
December 1 - 4, 2025
Hong Kong, Hong Kong

Conference Sponsors:
SIGCOMM

A Trustworthy and Comprehensive Communication Auditing System for High-Performance Gateways

Yu Jin
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, Jiangsu, china
jiny2048@gmail.com

Daming Huang
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, Jiangsu, china
huangdm@nju.edu.cn

Jincheng Xiang
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, Jiangsu, china
jinchengxiang@smail.nju.edu.cn

Chen Tian
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, Jiangsu, china
tianchen@nju.edu.cn

Abstract

With the rapid development of regulatory technology and industrial internet, communication auditing systems are facing increasing demands for performance, integrity, and trustworthiness. Traditional approaches often suffer from incomplete data capture, high risk of tampering, and poor scalability in high-concurrency environments. This paper proposes a high-performance and trustworthy communication auditing system tailored for edge gateways. The system leverages eBPF-based kernel-level traffic monitoring to achieve low-overhead and high-precision data collection, and introduces a hash tree aggregation algorithm to compress massive audit logs before blockchain submission. Key components are protected by Intel SGX to ensure secure processing, while the root hashes are immutably stored on a regulatory blockchain via smart contracts. The system supports real-time processing and sub-second querying over large-scale traffic logs, making it suitable for blockchain supervision, edge computing, and industrial control scenarios. Experimental results demonstrate that the proposed system significantly outperforms existing solutions in throughput, latency, and audit integrity, offering strong potential for practical deployment.

CCS Concepts

• **Security and privacy** → Domain-specific security and privacy architectures; • **Networks** → **Network monitoring**; • **Information systems** → **Data management systems**.

Keywords

Communication audit; eBPF; hash tree; blockchain; trusted execution; SGX; edge gateway; traffic monitoring

BlockNetSys '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2244-8/2025/12
https://doi.org/10.1145/3769698.3771229

ACM Reference Format:
Yu Jin, Daming Huang, Jincheng Xiang, and Chen Tian. 2025. A Trustworthy and Comprehensive Communication Auditing System for High-Performance Gateways. In *Proceedings of the 2025 ACM CoNEXT Workshop on Blockchain-Network Synergy (BlockNetSys '25), December 1–4, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3769698.3771229

1 Introduction

With the rapid development of network infrastructure and the migration of critical business systems to the cloud, the importance of network security and communication auditing has become increasingly prominent. In particular, in scenarios such as government regulation, financial auditing, and the industrial internet, the authenticity, integrity, and traceability of communication data have become essential components of compliance requirements. However, mainstream network auditing systems still rely heavily on traditional deep packet inspection (DPI) and peripheral log collection methods, which suffer from complex deployment, performance bottlenecks, and difficulties in achieving precise auditing of full-path and full-volume communication data.

To address these challenges, this paper proposes a trustworthy and comprehensive communication auditing system designed for high-performance gateways. The system integrates multiple advanced technologies including eBPF, zero-copy transmission, hash tree aggregation, trusted execution, and blockchain-based evidence preservation. It enables full-process, full-path, and full-field auditing of network communication behaviors, and provides reliable data support and online query capabilities for third-party regulatory platforms.

2 Background

Traditional network auditing systems typically adopt a centralized deployment architecture and complete data acquisition and analysis through traffic mirroring, passive packet capturing, and log merging[3, 6, 9–14]. These approaches face several common bottlenecks:

- **Performance bottleneck**: In high-frequency traffic scenarios, packet loss is severe, making it difficult to ensure completeness of audit data;

- **Insufficient trustworthiness**: Logs can be modified by local systems or third parties, making them unreliable for regulatory use;
- **Low query efficiency**: The lack of structured indexing and semantic abstraction limits real-time auditing and traceability.

In recent years, the rapid advancement of eBPF technology[15, 16] and blockchain-based evidence mechanisms has created new opportunities for building trustworthy auditing systems. eBPF allows flexible instrumentation within the kernel space to support high-performance traffic collection, while blockchain provides inherent immutability, making it ideal for preserving audit records. The combination of these technologies enables the construction of communication auditing systems that feature high throughput, low latency, strong trust guarantees, and high availability.

3 Related Work

Numerous studies have been conducted both domestically and internationally on network traffic auditing, data aggregation, and blockchain-based notarization, which can be categorized as follows:

- **DPI-based traffic auditing systems**: Open-source intrusion detection systems such as Bro/Zeek[8] and Suricata[7] rely on deep packet inspection and rule-based analysis of network behavior but struggle with performance in high-concurrency environments;
- **eBPF-based network observability platforms**: Projects such as Facebook Katran[5] and Cilium[2] utilize eBPF to implement high-performance load balancing and policy enforcement, with partial support for traffic sampling;
- **Data notarization and trusted storage systems**: Consortium blockchains such as Hyperledger Fabric[1] and FISCO BCOS[4] support high-frequency data writes, but still face throughput limitations;
- **Trusted computing protection mechanisms**: Hardware-based trusted execution environments like Intel SGX are widely used to securely isolate sensitive data processing, and are suitable for building trusted data acquisition and handling pipelines.

However, existing studies often focus on optimizing individual components and lack a unified design that covers the entire pipeline from data acquisition to aggregation, notarization, and query. This paper integrates these key technologies into a complete auditing architecture tailored for high-frequency edge communication scenarios, offering improved performance and enhanced trustworthiness.

4 System Design

The auditing system consists of a traffic capture module, a trusted data aggregation module, a regulatory database, and a secure regulatory blockchain. The system submits audit data to the secure blockchain for tamper-proof storage, while providing a dedicated query interface for third-party regulatory platforms, as shown in Fig. 1.

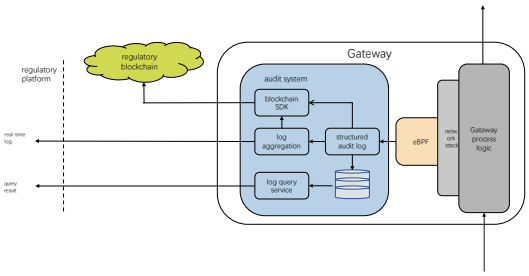

Figure 1: audit system architecture

4.1 Traffic Capture

4.1.1 High-Performance Traffic Capture Mechanism Based on eBPF. To meet the demands of high-throughput and low-overhead network monitoring, the system adopts eBPF technology in place of the traditional Netfilter/conntrack architecture. It implements a collaborative traffic capture mechanism between the kernel and user space. This mechanism supports real-time extraction of each TCP/UDP flow's five-tuple (source IP, destination IP, source port, destination port, protocol) along with its related state (packet count, byte count, flow start and end times), and efficiently transmits them to user space for upper-layer modules.

4.1.2 eBPF Program Design and Deployment. eBPF programs are attached to key points in the network data path (TC ingress and TC egress) to extract five-tuple information of each incoming packet in real time, which is then used as a key to construct flow identifiers. Each active flow maintains a state entry in an eBPF hash map (`BPF_MAP_TYPE_HASH`), including:

- **Packet count**: The total number of packets in the current flow;
- **Total bytes**: The cumulative payload size of received packets;
- **Flow start time**: Timestamp when the five-tuple was first observed;
- **Last active time**: Timestamp of the most recent packet, used for flow termination detection.

Timestamps are obtained via `bpf_ktime_get_ns()`. All state updates are performed atomically in the kernel to avoid extra overhead from user-space intervention.

4.1.3 Kernel-Space Log Buffer Design. To support efficient data export and state replay, the system implements a ring buffer structure entirely within the kernel. The log buffer consists of two types of eBPF maps:

- **Log entry area**: A `BPF_MAP_TYPE_PERCPU_ARRAY` map stores flow logs, ensuring each CPU core has an independent buffer to avoid write contention;

- **Control pointer area**: A standard `BPF_MAP_TYPE_ARRAY` map maintains the `head` and `tail` pointers for each log buffer, enabling ring structure write/consume management.

After capturing a new flow or updating an existing one, the eBPF program writes a log entry to the `tail` position of the corresponding per-CPU buffer and updates the control pointer. The log entry structure is as follows:

```
struct flow_log_entry {
    __u64 timestamp_ns;
    __u32 src_ip;
    __u32 dst_ip;
    __u16 src_port;
    __u16 dst_port;
    __u8  protocol;
    __u64 pkt_count;
    __u64 byte_count;
    __u64 flow_start_ns;
    __u64 flow_end_ns;
};
```

4.1.4 User-Space Consumption Mechanism. The user-space program extracts flow state information from the eBPF ring buffer using a hybrid approach combining polling and event-driven mechanisms, thereby maintaining comprehensive visibility over active and historical flows. Inspired by the NewAPI model, the system dynamically switches consumption modes based on request intensity to balance performance and resource utilization.

- **Polling Mode**: Under high system load or frequent user-space processing, the program periodically polls `head` and `tail` pointers to batch extract unconsumed log entries.
- **Event-Driven Mode**: When request pressure is low or the system is idle, the mechanism switches to an event-driven mode. The kernel pushes critical log entries to a perf ring buffer via `bpf_perf_event_output()`, where they are asynchronously received and processed by user space.

The system continuously monitors log backlog, consumption frequency, and resource usage to dynamically select the optimal consumption strategy. This reduces unnecessary polling overhead while maintaining processing capability. The log buffer also supports persistent storage: user-space programs can periodically flush logs to disk, and replay them during recovery to reconstruct flow states, enhancing system stability and auditability.

4.1.5 Performance Advantages and Scalability. This capture mechanism offers the following advantages:

- **Low Overhead**: eBPF programs run in kernel space, avoiding frequent context switches;
- **High Concurrency**: per-CPU maps enable lock-free concurrent writes, supporting multicore environments;
- **High Reliability**: Persistent logging and replay ensure data completeness;
- **Strong Scalability**: The structure is flexible and easily extensible for new fields or policies.

This design provides strong foundational support for gateway deployment scenarios and is widely applicable to high-frequency communication, blockchain consensus synchronization, and other use cases demanding high monitoring precision and throughput.

4.2 Trusted Aggregation

The gateway uses a high-performance network server to handle terminal requests and supports hundreds of thousands of concurrent blockchain submission requests via numerous lightweight user-space processing handles. Each request generates a regulatory log entry. However, uploading all raw logs in real time exceeds the throughput capability of current high-performance consortium blockchains. Therefore, the system aggregates and compresses raw logs in batches before submitting the results to the regulatory blockchain.

Data extracted from the eBPF ring buffer first enters a processing queue, where each log is hashed using an AVX512-optimized SHA-256 algorithm for high-performance computation. After processing, each log entry is standardized into a unified structure: a triplet of (log, index, hash). These triplets are uploaded to a high-performance distributed database, with indexes built on both the index and hash fields to support subsequent data updates.

Simultaneously, the logs are sent in parallel to a data aggregation program. To maximize processing speed, the interface between the processing queue and the aggregator is implemented as a content-sharing ring buffer. The queue writes transaction hashes to the buffer, while the aggregator reads them for data aggregation. Several optimizations are applied to enhance inter-process communication:

- Multiple ring buffers are used for parallel data writing to fully utilize multithreading and increase processing concurrency;
- Serialization/deserialization is avoided: data is written to the ring buffer directly according to a predefined memory layout, eliminating the CPU overhead of these processes;
- The data aggregator adopts a NAPI-inspired strategy to optimize CPU usage: when data is written frequently, polling is used to read the latest entries; when data arrival is sparse, signals are used to notify the aggregator to read new entries.

The aggregation program builds a hash tree from the collected data. To fully leverage multithreading, multiple worker threads concurrently consume ring buffer entries and write them to local leaf nodes of their respective hash trees. A worker thread stops consuming entries when either a timeout is reached or its leaf node is full, at which point it hands off the ring buffer to a new thread and starts computing hash values from the leaf node up to the root. This process is detailed in Algorithm 1.

To ensure data integrity and prevent tampering, the aggregation program adopts a blockchain-inspired chaining mechanism: each root hash is linked to previous root hashes, forming an irreversible verification chain. However, since multiple worker threads simultaneously compute and construct hash trees, naively chaining root hashes linearly would cause contention. To avoid this, we improve the linking method by allowing each new root hash to link to any previous unlinked root hashes. This allows all threads to link hash trees without blocking, forming a tamper-proof Directed Acyclic Graph (DAG).

After completing a round of processing, each worker thread returns its hash tree and corresponding root hash to the audit system. The system then parses the tree and updates the database

Algorithm 1 hash tree aggregate algorithm

Input: $RingBuffers$
$MaxLeaves$: max leaves of a tree (default 32768)
$Timeout$: batch timeout (default 100ms)
Output: $Forest$: generated hash tree

init global hash chain $Chain \leftarrow \emptyset$
create work thread pool $Workers[1..N]$
allocate private mempool for each worker $MemPool_i$
for each worker $W_i \in Workers$ **do**
  while system running **do**
    $LocalLeaves \leftarrow \emptyset$
    $Timer \leftarrow Time.Now()$
    while $|LocalLeaves| \leq MaxLeaves$ **and** $Time.Now() < Timer + Timeout$ **do**
      read hash from $RingBuffers$
    end while
    build hash tree T:
    for $level \leftarrow 1$ **to** $\log_2(MaxLeaves)$ **do**
      for $j \leftarrow 0$ **to** $|Nodes_{level-1}|/2$ **do**
        $parentHash \leftarrow \text{SHA256}(Nodes_{level-1}[2j] \parallel Nodes_{level-1}[2j+1])$
        $Nodes_{level} \leftarrow Nodes_{level} \cup \{parentHash\}$
      end for
    end for
    $rootHash \leftarrow Nodes_{top}[0]$
    if $Chain \neq \emptyset$ **then**
      $chainedRoot \leftarrow$ SHA256([For chainedLeaf in Chain] $\parallel$ $rootHash$)
    else
      $chainedRoot \leftarrow rootHash$
    end if
    $Chain \leftarrow Chain \cup \{chainedRoot\}$
    update DB
  end while
end for

for each transaction (storing hash, root hash, and position within the tree), and saves the entire hash tree to a local database.

The aggregation program has both a standard and a trusted computing version. In the trusted version, the entire aggregation process is protected using Intel SGX to ensure execution integrity.

4.3 Trusted Evidence Storage

For trusted evidence storage and regulatory auditing, key features of each network flow are extracted to form high-level semantics. After computing the hash tree, the root hash is submitted to a secure regulatory sidechain via smart contracts for permanent storage, while the high-level semantics are transmitted in real time to the regulatory platform through a dedicated network. The smart contract implements a robust anti-replay mechanism and timestamping service, ensuring the immutability of audit records.

During the verification phase, regulators or users can confirm the integrity of the original audit logs by locally computing the hash of the raw log data and comparing it with the on-chain evidence. Matching results indicate that the original logs have not been tampered with.

4.4 Online Query

For log processing and storage, traffic log data is transferred from kernel space to user space using a zero-copy mechanism, then processed through multi-level buffering and batch operations before being persistently stored in a high-availability, high-performance database. The storage system employs mechanisms such as Write-Ahead Logging (WAL) to ensure data reliability. The database tables are specifically designed with optimized index structures for network flow characteristics, enabling the regulatory platform to filter and query records based on attributes such as IP address, time range, and geographic location.

5 Experimental Evaluation

To verify the system's performance and usability in practical environments, we deploy the proposed solution on real-world edge nodes and evaluate key performance metrics, including traffic capture throughput, log aggregation latency, and blockchain write efficiency.

5.1 Experimental Setup

The experimental platform consists of one control node and four gateway nodes, each equipped with an Intel Xeon Silver 4214 CPU (16C/32T) and 256GB of RAM. The control node runs the regulatory platform and blockchain network, while the gateway nodes host the auditing system components. In the experiments, 500,000 concurrent connection requests are simulated, with a log capture cycle of 100ms and a maximum of 32,768 log entries per aggregation batch.

5.2 Throughput Performance

Under high-concurrency request loads, the eBPF module achieves an average capture throughput of 700 Mbps. Each gateway node collects over 500,000 log entries per second, while the overall CPU utilization remains below 35%.

5.3 Aggregation Latency

Under typical system load, the average latency from log capture to aggregation completion is approximately 185ms. The aggregation process benefits from multithreaded optimization and AVX acceleration to reduce computational overhead. With the NAPI-inspired mode-switching strategy, latency drops to as low as 40ms under light-load conditions.

5.4 Attestation Efficiency

The test blockchain is based on the FISCO BCOS consortium chain, supporting approximately 600 aggregated transactions per second. After hash tree-based compression, the system can attest up to 19 million raw log entries per second, effectively meeting the demands of medium to large-scale government and enterprise regulatory scenarios.

6 Conclusion

This paper presents the design and implementation of a trustworthy and comprehensive communication auditing system tailored for high-performance gateways, covering the full auditing pipeline including traffic capture, trusted aggregation, on-chain attestation, and online query. By integrating technologies such as eBPF, hash tree aggregation, SGX, and blockchain, the system achieves high-throughput, low-latency, and trustworthy network auditing capabilities, making it suitable for critical application scenarios such as edge computing, consortium blockchain regulation, and industrial control.

Experimental results demonstrate that the system exhibits strong potential for real-world deployment in terms of both performance and reliability.

Future work will focus on enhancing the system's support for complex application-layer protocols, incorporating machine learning-assisted analysis for automated auditing and anomaly detection, and exploring multi-chain attestation and cross-domain collaborative auditing mechanisms to support large-scale distributed regulatory scenarios.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No. 2022YFB2702800.

References

[1] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference* (Porto, Portugal) *(EuroSys '18)*. Association for Computing Machinery, New York, NY, USA, Article 30, 15 pages. doi:10.1145/3190508.3190538

[2] The Cilium Authors. 2025. *cilium*. https://cilium.io/

[3] Huayi Duan, Yuefeng Du, Leqian Zheng, Cong Wang, Man Ho Au, and Qian Wang. 2023. Towards Practical Auditing of Dynamic Data in Decentralized Storage. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 708–723. doi:10.1109/TDSC.2022.3142611

[4] FISCO. 2016. *FISCO*. http://fisco-bcos.org/

[5] Inc GitHub. 2025. *katran*. https://github.com/facebookincubator/katran

[6] Haoxiang Han, Shufan Fei, Zheng Yan, and Xiaokang Zhou. 2022. A survey on blockchain-based integrity auditing for cloud data. *Digital Communications and Networks* 8, 5 (2022), 591–603. doi:10.1016/j.dcan.2022.04.036

[7] OISF. 2025. *suricata*. https://suricata.io/

[8] The Zeek Project. 2024. *Zeek*. https://zeek.org/

[9] Liping Qiao, Yanping Li, Feng Wang, and Bo Yang. 2022. Lightweight integrity auditing of edge data for distributed edge computing scenarios. *Ad Hoc Networks* 133 (2022), 102906. doi:10.1016/j.adhoc.2022.102906

[10] Hamed Tabrizchi and Marjan Kuchaki Rafsanjani. 2020. A survey on security challenges in cloud computing: issues, threats, and solutions. *The Journal of Supercomputing* 76, 12 (01 Dec 2020), 9493–9532. doi:10.1007/s11227-020-03213-1

[11] Jun-Feng Tian and Hao-Ning Wang. 2020. An efficient and secure data auditing scheme based on fog-to-cloud computing for Internet of things scenarios. *International Journal of Distributed Sensor Networks* 16, 5 (2020), 1550147720916623. doi:10.1177/1550147720916623 arXiv:https://doi.org/10.1177/1550147720916623

[12] Jiaojiao Wu, Yanping Li, Fang Ren, and Bo Yang. 2021. Robust and auditable distributed data storage with scalability in edge computing. *Ad Hoc Networks* 117 (2021), 102494. doi:10.1016/j.adhoc.2021.102494

[13] Chuan Zhang, Haojun Xuan, Tong Wu, Ximeng Liu, Guomin Yang, and Liehuang Zhu. 2024. Blockchain-Based Dynamic Time-Encapsulated Data Auditing for Outsourcing Storage. *IEEE Transactions on Information Forensics and Security* 19 (2024), 1979–1993. doi:10.1109/TIFS.2023.3338485

[14] Yu Zhao, Yangguang Tian, Chunbo Wang, Xiaoqiang Di, and Hui Qi. 2024. Smart Contract-Based Auditing of Edge Data for Vehicular Networks. In *2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2045–2050. doi:10.1109/TrustCom63139.2024.00284

[15] Yang Zhou, Zezhou Wang, Sowmya Dharanipragada, and Minlan Yu. 2023. Electrode: Accelerating Distributed Protocols with {eBPF}. 1391–1407. https://www.usenix.org/conference/nsdi23/presentation/zhou

[16] Yang Zhou, Xingyu Xiang, Matthew Kiley, Sowmya Dharanipragada, and Minlan Yu. 2024. DINT: Fast In-Kernel Distributed Transactions with eBPF. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)* (Santa Clara, CA, 2024-04). USENIX Association, 401–417. https://www.usenix.org/conference/nsdi24/presentation/zhou-yang