Blockchain-based Secure Log Management with LLM-assisted Querying System

Warayut Saena[1], **Chayada Surawanitkun**[1], **Manussawee Nokkaew**[1], **Kanda Sorn-In**[1], **Nongram Mueanrit**[1], and **Satit Kravenkit**[2†], Non-members

ABSTRACT

This research presents a blockchain-based log management system that integrates an LLM-assisted query mechanism to enhance security and provide contextual insights into storage event logs. The modules for decentralized storage event logs, including registered users, log retrieval and storage, and log auditing, are built on a blockchain deployed via an Ethereum smart contract, which enhances the following. EdDSA-based digital signatures (Ed25519) are applied to audit log events to ensure integrity. Additionally, log files are encrypted using asymmetric encryption (Curve25519) to protect confidentiality, while the decentralized IPFS (InterPlanetary File System) stores log file metadata on the blockchain, reducing storage size. In the querying module, an LLM-based natural-language model is integrated. Moreover, we organize the logs into semantic groups and hourly summary tables for LLM-assisted analysis. Users can analyze log data without complex query rules. The experiment covers five evaluations: First, functional correctness. The storage event log modules correctly support secure log management and verify log audits. Second, IPFS delivers practical performance with average upload times (0.41 s) and download times (0.19 s). Third, blockchain operations require reasonable gas usage. Fourth, the Llama 3.1 (8B) achieves the best querying performance, with correctness (0.80), faithfulness (0.53), and helpfulness (0.67) supporting practical log analysis. Fifth, hourly summarization improves query efficiency by reducing average query time from 0.57 ms to 0.38 ms.

Keywords: Event Log, Decentralized Storage, Blockchain Technology, InterPlanetary File System (IPFS), Large Language Model (LLM)

1. INTRODUCTION

The storage and verification of event logs are essential parts of information security systems at organizational and global levels. Data protection laws and standards, including the General Data Protection Regulation (GDPR) [1], NIST Cybersecurity Framework [2], and ISO/IEC 27001 [3], require organizations to audit access, track changes, and review historical events. This allows organizations to verify investigations of security incidents. Therefore, critical protection laws and standards mandate the creation of a log management system that ensures integrity, availability, and traceability.

Log data [4–6] records activities and events in network devices, such as network communication, server operations, and firewall events. Log attributes typically include event id, packet identifiers, IP addresses, ports, protocols, and status messages, and interpreting these fields requires technical expertise [7]. Most current log storage systems still utilize centralized logging architecture [8], which introduces several security limitations. A failure at the central log server renders the entire log storage system inaccessible or results in data loss. Additionally, centralized storage increases the risk of data tampering and decreases auditability.

Moreover, cloud systems generate large amounts of log data through continuous monitoring of user access, service requests, and system health across distributed infrastructures [4]. Similarly, IoT devices generate extensive logs by sending sensor readings and device status updates from connected endpoints [9]. The large volume of log data can create bottlenecks in centralized systems and reduce system performance [8].

To overcome the limitations of centralized systems, a decentralized event-log storage architecture is a practical solution. Blockchain, a decentralized technology [6], provides a distributed ledger maintained by multiple nodes. It reduces reliance on a single server and lowers the risk of data loss or unauthorized modifications. Most blockchain systems use the InterPlanetary File System (IPFS) [10] for decentralized file storage. IPFS identifies data using an IPFS address rather than a fixed storage location [11], allowing files to be distributed across multiple nodes.

Combining blockchain and IPFS in decentralized storage models [10] is effective. The blockchain records the hash values of data in a sequence of linked blocks [7] and thus allows clear and reliable verification of past records. Event log files are distributed across multiple

Manuscript received on December 25, 2025; revised on July 8, 2026; accepted on July 24, 2026. This paper was recommended by Associate Editor Chanon Warisarn.

[1]The authors are with Center of Multidisciplinary Innovation Network Talent (MINT Center), Faculty of Interdisciplinary Studies, Khon Kaen University, Nong Khai Campus, Nong Khai, Thailand.

[2]The authors are with Department of Computer Science, College of Computing, Khon Kaen University, Khon Kaen, Thailand .

[†]Corresponding author: satikr@kku.ac.th

©2026 Author(s). This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License. To view a copy of this license visit: https://creativecommons.org/licenses/by-nc-nd/4.0/.

Digital Object Identifier: 10.37936/ecti-eec.2026243.262937

nodes in the IPFS peer-to-peer network [10] to improve system availability and reduce the risk of a single point of failure. The blockchain stores the hash values of log data, allowing any modification of the log data to be detected through hash verification.

Traditional storage in the log querying system is used to collect and store logs from systems and network devices on a central server. Log analysis commonly relies on conditional queries or Regular Expressions (Regex) to search log files [12]. This approach is suitable for small log volumes. However, query performance decreases as log volume grows. Most studies [4], [7] focus on log storage and integrity verification but provide limited support for efficient querying of large-scale log data.

This paper presents a secure decentralized log management and querying framework that uses blockchain and IPFS to store and verify firewall log records, ensuring audit transparency. Smart contracts support log registration, storage, retrieval, and counting. Digital signatures and encrypted log storage are integrated to verify source authenticity and maintain data integrity. In addition, a Large Language Model (LLM) supports natural language querying. Event logs are organized into behavioral groups and hourly summaries to improve query efficiency. A case study using real university firewall logs demonstrates that the proposed framework simplifies log analysis and reduces response time.

2. LITERATURE REVIEW

In this section, we review research on the management and analysis of log events. First, decentralized log storage using blockchain and IPFS is examined to address single-point failures, data tampering, and incident auditing. Second, the use of LLMs is investigated to analyze log events in natural language. Third, cryptographic integrity is reviewed to examine the public-private key cryptography solution. Lastly, research limitations are identified.

2.1 Decentralized Log Storage

Centralized log systems remain vulnerable to single-point failures, increasing the risk of log data loss and unauthorized modification if the central storage is compromised or becomes unavailable [4], [7, 8]. To address these risks, decentralized log storage based on blockchain and IPFS has been proposed [13–15]. Blockchain prevents unauthorized modification by recording log metadata as immutable, traceable transactions, enabling integrity verification and post-incident auditing. In parallel, IPFS mitigates data loss from single-node failures by distributing encrypted log files across multiple nodes, thereby eliminating reliance on a single storage server.

2.2 Log Querying and Summarization LLM

Current log querying and analysis remains challenging because log events are generated continuously in varying formats and are difficult to analyze in large quantities. Most log querying tools use regular expressions (regex) or rule-based techniques to parse log events, which work well for simple pattern matching but are inadequate for natural language queries and context-driven log summarization [16, 17]. To address these limitations, Large Language Models (LLMs) have been employed for log-related analytical tasks that require contextual understanding of security information and log events. Recent studies have applied LLMs to log parsing, anomaly detection, fault diagnosis, and log summarization, demonstrating improved accuracy in event interpretation compared to traditional rule-based approaches [4].

2.3 Cryptographic Integrity and Key-Based Verification

Verifying the data source and completeness is essential in event log recording to prevent modification during transmission or storage. Public-private key cryptography, also known as asymmetric cryptography, is commonly used to ensure data authenticity and integrity by signing log data with a private key and verifying it with the corresponding public key [18–20]. Several studies [7, 8], [13, 14] further show that combining public-private key cryptography with blockchain improves the reliability of log data, as blockchain provides immutable storage that prevents post-recording tampering of signed log entries.

2.4 Research Limitation

The proposed decentralized logging systems, which integrate blockchain technology and IPFS, primarily store log files on IPFS and record metadata on the blockchain, but encryption and identity verification are not sufficiently emphasized [9]. Therefore, we develop Ethereum smart contracts to manage user permissions. We also integrate digital signatures to verify source authenticity and use an encrypted log to prevent data tampering.

Moreover, recent studies have applied LLMs to log analysis systems for anomaly detection and log interpretation [4], [16]. However, the explicit log structuring and summarization strategies designed to improve LLM efficiency are rarely discussed. Therefore, we proposed transforming event logs into behavior-based, hourly summaries before the LLM to improve query efficiency and accuracy.

Table 1 compares the proposed framework with representative studies related to blockchain-based log management and LLM-based log analysis. The comparison focuses on blockchain/IPFS integration, encryption and digital signature support, LLM-based querying, and log summarization.

The remainder of this paper is organized as follows. Section 3 presents the research methodology. Section 4 describes experimental data collection and analysis. Section 5 presents the results, and Section 6 concludes the paper.

***Table 1**: Comparison of Existing Log Management Frameworks.*

Existing Work	BC/ IPFS	Enc./ Sign.	LLM Query	Log Sum
Li et al. [7]	✓	✓	×	×
Rakib et al. [8]	✓	✓	×	×
Jain et al. [13]	✓	✓	×	×
Guan et al. [16]	×	×	✓	×
Proposed Framework	✓	✓	✓	✓

3. METHODOLOGY

The methodology of the proposed framework is presented through the architectural design, operational workflow, and system enhancements. An overview of the framework architecture is shown in Fig. 1. The methodology comprises three parts, including the system components described in Section 3.1, the system workflow in Section 3.2, and the system enhancements and contributions in Section 3.3.

3.1 System Components

The proposed framework consists of eight components, each performing a specific function related to event log collection, secure storage, integrity verification, auditing, and query-based analysis in a decentralized environment. The role of each component is as follows.

(1) The firewall monitors network traffic inside and outside the organization and serves as the event log source. An admin can assign rules-based security that allows or denies events. All firewall events are compressed to the `.csv.zst` format to reduce storage usage.

(2) The Log Manager is a central server that receives event logs from the firewall and prepares them for secure decentralized storage. The primary tasks involve obtaining log files and generating EdDSA digital signatures, which can be utilized to verify the source logs' authenticity and integrity. The logs are encrypted using GPG (GNU Privacy Guard) to ensure the confidentiality of the log files. The encrypted log files are transmitted to IPFS. The log metadata is then recorded on the blockchain.

(3) IPFS decentralized storage is used to store encrypted log files. When a log file is uploaded, IPFS automatically generates a unique IPFS address from its contents. This IPFS address is then recorded on the blockchain as part of the log metadata.

(4) Smart contracts are responsible for the log storage management system, which mainly consists of 3 modules, including, firstly, user registration and role assignment, which define authorized users. Secondly, log counting provides an overview of stored log records. For the last log retrieval, the log metadata stored on the blockchain is used as a reference to identify and locate the corresponding encrypted log files.

(5) The off-chain database based on DuckDB stores decrypted log files in a tabular format, where each event log record is stored as a row, and log attributes are stored as columns. This structure allows efficient filtering and aggregation during analysis. To improve query efficiency, the proposed system generates hourly summaries by grouping log events into fixed one-hour time windows. This representation reduces the amount of data processed during LLM-based log analysis.

(6) The LLM-based query processing module acts as the user-facing interface of the system and operates an LLM deployed within the organization. The module uses an LLM to generate SQL [16] commands from natural-language user queries for database execution and to produce human-readable summaries of retrieved log data.

(7) The security audit component supports inspection of log access and system security. Access permissions are validated based on the auditor's blockchain address recorded in the smart contract. The integrity and origin of log files are verified using EdDSA digital signatures [20]. Authorized auditors can decrypt the log files and inspect their contents for compliance verification.

(8) Users are defined in two roles. First, network administrators interact with the system through the LLM module to perform routine log queries, monitor network behavior, and analyze operational events. Second, security auditors are responsible for audit-related tasks, as described in (7).

3.2 System Workflow

This section presents the workflow for the decentralized log management and querying system, from collecting event logs from the firewall through to LLM-based querying.

Firstly, (Component 1) the firewall generates security event logs, which are compressed into `.csv.zst` files to reduce data volume. (Component 2) The Log Manager periodically retrieves the log files and prepares them for secure storage (Step 3.2.1). Then, the system applies a digital signature using EdDSA to ensure the integrity of the event log files and encrypts them with GPG to protect their confidentiality (Step 3.2.2).

Next (Component 3), the encrypted log files are uploaded to IPFS, which generates an IPFS address that serves as a reference to the stored files. The corresponding log metadata, including the blockchain address, public key, timestamp, and IPFS address, is recorded in a smart contract on the Ethereum blockchain (Component 4) for integrity verification and auditing purposes (Step 3.2.3).

When Component 5 (DuckDB) retrieves event logs, DuckDB sends a request to the smart contract to obtain the required log metadata (Step 3.5.1). The smart contract verifies the DuckDB access permissions and returns the corresponding metadata, including the IPFS address (Step 3.5.2). Then, the returned metadata is used by DuckDB to locate and download the encrypted log file from IPFS (Step 3.5.3).

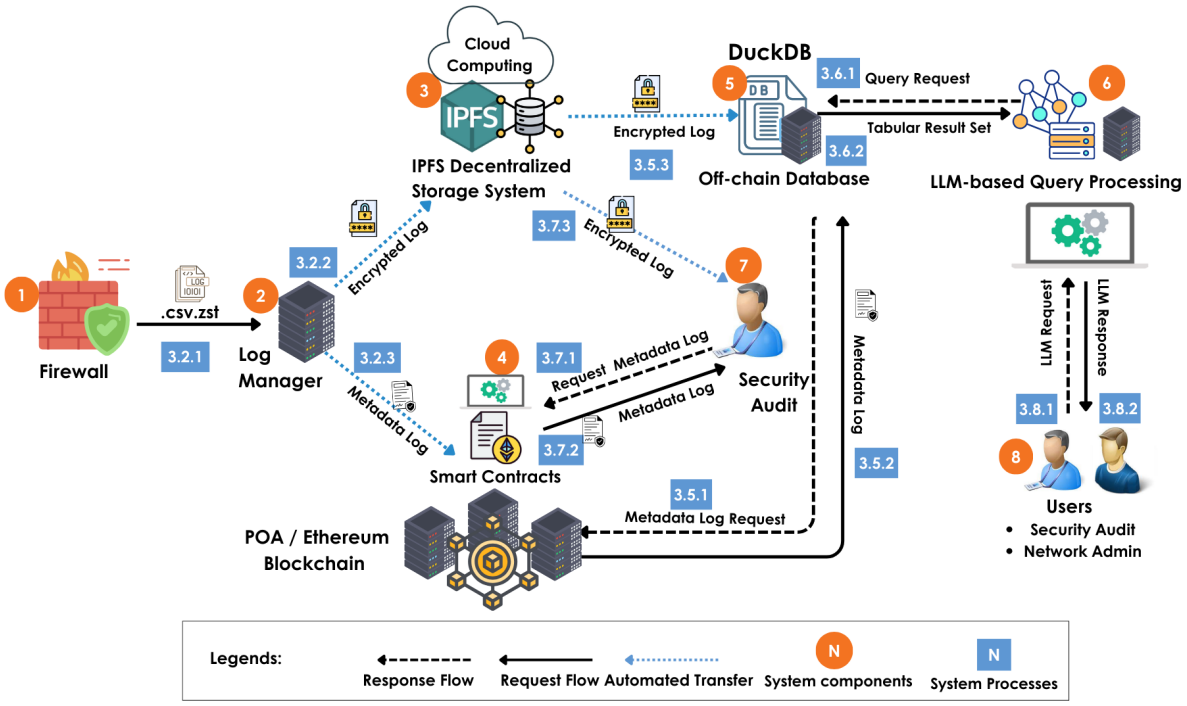

***Fig. 1**: Framework Storage and Analysis of Event Logs Query system.*

The Log Manager performs log decryption before the event logs are imported into DuckDB. The corresponding private keys are securely stored on the authorized local server. DuckDB operates within a trusted internal environment with access restricted to authorized users. After the event logs are imported into DuckDB, the Log Manager groups log records into hourly summaries to reduce data volume and improve query efficiency. In addition, a descriptive `summary_text` field is generated to support efficient LLM-based querying and analysis, as described in detail in Section 3.3.2. The hourly summaries are automatically updated after new log files are imported into DuckDB.

The LLM-based querying module (Component 6) then receives a natural-language question, and the system formulates a database query for execution in DuckDB (Step 3.6.1). The proposed framework does not use a dedicated text-to-SQL library. Instead, SQL queries are generated through prompt-based interactions with the LLM using database schema information and summarized log context. In addition, generated SQL statements are validated before execution to reduce invalid or unsafe queries. The system selects log records based on the user query and the related metadata stored on the blockchain. For large datasets, the system first searches the hourly summaries table to reduce the search scope before retrieving the related raw log records from DuckDB and IPFS. The next step applies a two-stage retrieval mechanism on the DuckDB server to improve query efficiency. In the first stage, the DuckDB server executes the query on the hourly summaries table to reduce the search scope. In the second stage, if the required information is not available in the summaries, the DuckDB server executes the query against the raw log tables. Then the database returns a tabular result set, a structured table composed of rows and columns that represent the query results (Step 3.6.2).

Moreover, the proposed system supports in-depth investigation through the security auditor role (Component 7). The auditor can request log metadata from the smart contract (Step 3.7.1). Using the returned IPFS address, the encrypted log file is retrieved from IPFS (Step 3.7.2) and examined in detail in accordance with audit permissions (Step 3.7.3).

For general operational queries or additional information requests (Component 8), users can submit questions via the LLM interface (Step 3.8.1), and the LLM will respond (Step 3.8.2).

3.3 System Enhancements and Contributions

To provide more detailed explanations of the proposed framework, this section describes two mechanisms that enable effective system operation: (1) the encryption process and log verification, and (2) behavioral grouping with hourly behavioral summarization.

3.3.1 Encryption Process and Log Verification

The proposed system employs log encryption and log verification to enhance security and trust in event log management. Log encryption and log verification are implemented through two main processes.

First, the encryption process protects the confidentiality of event logs during storage and transmission. The proposed system uses GPG (OpenPGP) as the crypto-

graphic framework for encryption and digital signing operations, encrypting event logs with a Curve25519 (X25519) public key and decrypting them with the corresponding private key. In this work, event logs are encrypted before being uploaded to IPFS, ensuring that unauthorized parties cannot access their contents despite IPFS's publicly addressable content identifiers. Each encrypted log file is associated with an IPFS address, which is then recorded on the blockchain as immutable metadata for secure and traceable storage management.

Log verification ensures the integrity and authenticity of event logs. The system uses the Ed25519 digital signature algorithm, a member of the EdDSA family, for digital signature verification through the GPG framework. This technique provides cryptographic security and performs comparably to RSA-2048 in terms of verification. Prior to transmitting encrypted log files to IPFS, they are digitally signed using an Ed25519 private key generated by the Security Audit Component. The blockchain stores a corresponding public key. The stored public key is employed to verify the digital signature of the log files obtained from IPFS, ensuring the authenticity of the log files. This enables auditors to verify the integrity and authenticity of the log files.

3.3.2 Behavioral Grouping and Hourly Behavioral Summarization

To support querying and behavior analysis, the proposed framework applies a behavioral grouping method with hourly summarization that classifies events based on security-related fields, including type, subtype, action, and status, found in event logs. The system analyzes these fields to assign events to categories. For example, events with action = login and status = failed are placed in the "AuthFail" group, while events related to intrusion detection are placed in the "IPS/Attack" group. Table 2 presents the basic grouping rules used in the framework.

Then, the LLM querying system produces hourly summaries by grouping log records using `device_id` and `hour_bin`. For each group, the system computes the total number of events (`event_count`) and identifies the most frequent source and destination IP addresses (`top_source_ip` and `top_destination_ip`). In addition, a `summary_text` field is generated to provide a readable description of the events, enabling the LLM to better understand the context. The structure of the hourly summary records is shown in Table 3.

The `summary_text` field uses a fixed format to convert summary data into readable sentences for semantic search. An example is shown below:

```
FG6H0FTB24900886 | 2025 07 06 00:00:
vpn/negotiate=8, vpn/ssl alert=5,
system/login=2, top_srcip: 77.83.207.136.
```

The hourly summary improves event analysis by reducing data volume and providing clearer contextual information.

***Table 2**: Semantic Event Categories.*

Type	Classification Criteria
Operational	General events with status = success
AuthFail	action = login and status = failed
TrafficDenied	action = deny
IPS/Attack	subtype related to IPS
VPN Activity	subtype = vpn

***Table 3**: Structure of Hourly Summary Records.*

Field	Description
device_id	The device identifier generates the log.
hour_bin	The hourly time window groups log events by hour.
event_count	The event count represents the number of events in each category.
top_source_ip	The top source IP indicates the most frequent source IP address.
top_destination _ip	The top destination IP indicates the most frequent destination IP address.
summary_text	The summary text provides context for the LLM.

4. EXPERIMENT

This section describes the experimental procedure used to assess the proposed framework, including the collection of real-case firewall logs, the definition of evaluation questions, evaluation methods and metrics, and the experimental setup.

4.1 Data Source and Collection Procedure

The experimental data were obtained using a FortiGate 600F firewall installed at the Information Technology Center of Phuket Rajabhat University. Log events were collected over a three-month period from September 1, 2025, to November 1, 2025, to demonstrate variations in system usage between school days and workdays.

The collected dataset contained 115 `.csv.zst` files of firewall event records. The most critical fields in a log include date, time, event type, subtype, action, source and destination IP addresses, status, service, and log description. The dataset illustrates both typical system activity and anomalous events, such as unsuccessful logon attempts and external intrusion attempts.

4.2 Definition of Evaluation Questions

To evaluate the system's ability to answer natural-language questions about event logs, 60 sample questions were collected from the university's network administrators. The questions represent real operational scenarios,

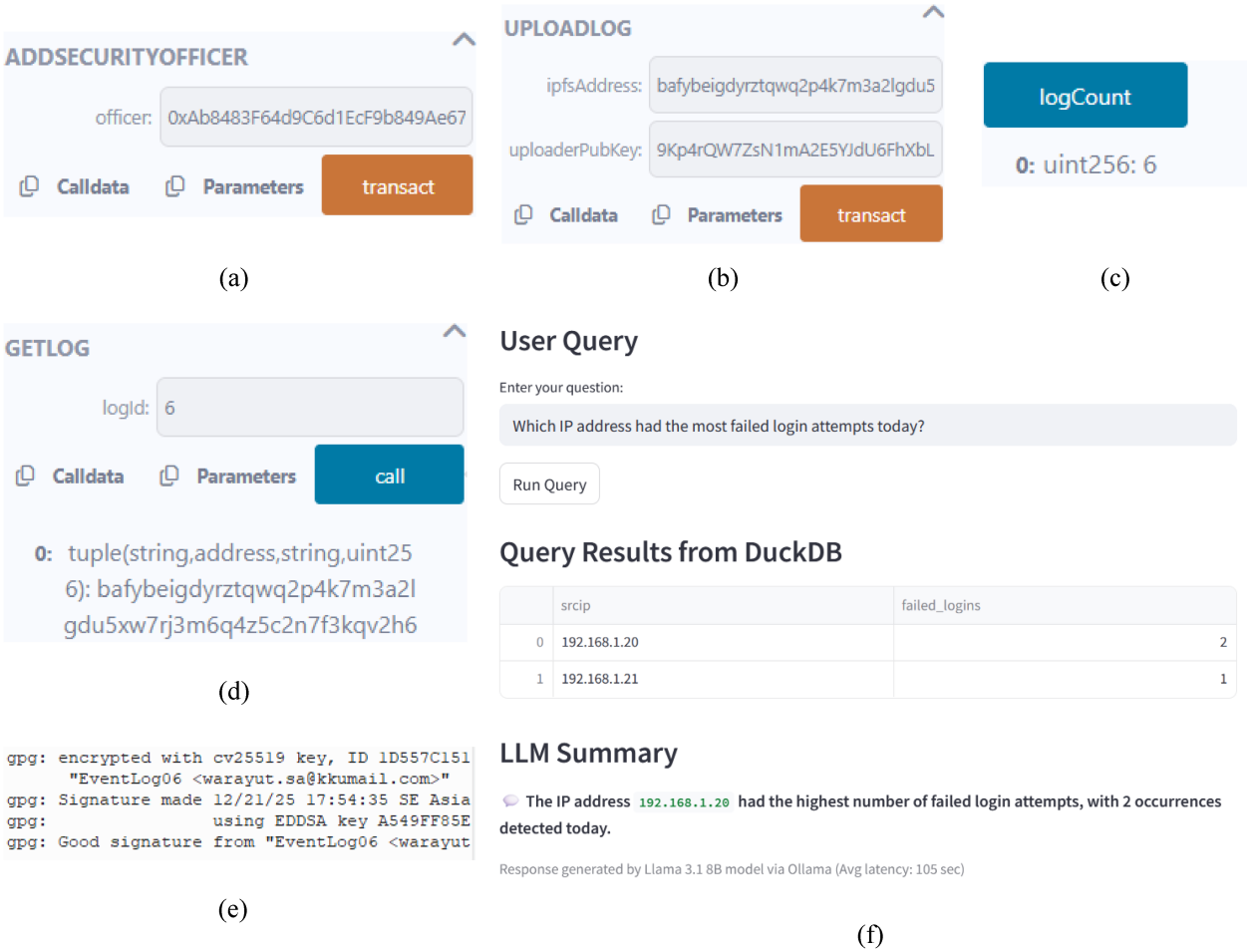

***Fig. 2**: Experiment Scenario.*

including login failure verification and identification of source IP addresses associated with abnormal behavior.

The questions were then grouped into four query intents that represent common patterns of log investigation in daily operations. First, count questions that request the number of events under specific conditions. Second, sequence questions examine the order or timeline of events. Third, attack questions focus on security incidents or intrusion attempts. Lastly, search questions aim to retrieve details from the event logs. Moreover, to increase the diversity and difficulty of the evaluation, the original question set was expanded from 60 to 200 items. These questions were synthesized using a few-shot prompting technique, following prior work on automatic question generation [17].

4.3 Evaluation Method

The framework evaluation covered five main aspects: functional correctness, IPFS upload and download latency, blockchain transaction and execution cost, LLM Performance and Quality, and hourly summarization-based query efficiency. The details are as follows:

(1) **Functional Correctness**: This metric evaluates the correctness of the proposed framework functions. Our evaluation covers four criteria: (1) event log encryption, (2) digital signature generation and verification, (3) log metadata recording on the blockchain, and (4) log retrieval and summarization using LLM.

(2) **IPFS Upload and Download Latency**: The upload and download times for log files of different sizes transferred through IPFS are measured to evaluate the data transfer performance under varying file sizes.

(3) **Blockchain Transaction and Execution Cost**: This metric evaluates gas usage for smart contract operations related to user registration, log metadata storage, and retrieval. The results indicate the cost of blockchain operations measured by gas usage.

(4) **LLM Performance and Quality**: The performance measurement evaluates the response latency from receiving a user question to producing the final answer. The quality of the answers is assessed using the LLM-as-a-Judge framework with three indicators: Correctness (C) in (1), Faithfulness (F) in (2), and Helpfulness (H) in (3), measuring content accuracy, consistency with log evidence, and practical usefulness of the answers. The LLM-as-a-Judge scores are computed as the average over all evaluation questions, as shown in Equations (1)–(3). This evaluation design is used to assess LLM-based

question answering in event log analysis tasks [21].

Correctness

$$C = \frac{1}{N}\sum_{i=1}^{N} c_i \tag{1}$$

Faithfulness

$$F = \frac{1}{N}\sum_{i=1}^{N} f_i \tag{2}$$

Helpfulness

$$H = \frac{1}{N}\sum_{i=1}^{N} h_i \tag{3}$$

where N is the number of evaluation questions, c_i, f_i, and h_i represent the scores assigned by the evaluation model to the i-th answer for correctness, faithfulness, and helpfulness, respectively, with values ranging from 0 to 1.

(5) **Query Efficiency with Hourly Summarization**: This metric evaluates query efficiency using hourly summarization. In DuckDB, the execution time of an aggregate query for counting security event records is measured on both the raw log and hourly summary tables. The query execution is repeated 20 times, and the average query execution time is calculated for comparison.

4.4 Experimental Setup

The framework was implemented on Ubuntu 24.04.3 LTS (x86_64) on a machine equipped with an 8-core CPU, 16 GB of RAM, and a 120 GB SSD, representing the environment of a medium-sized university organization. As shown in Fig. 1, this single machine hosts the Log Manager (Component 2), the DuckDB-based log processing module (Component 5), and the LLM-based querying module (Component 6).

The IPFS component runs on a local IPFS node, while the blockchain component operates on a local testnet built on an EVM-compatible platform. For log processing, DuckDB is used as the OLAP engine and is connected to the Ollama platform to run LLM models, including Qwen2.5:3B, DeepSeek R1:7B, Gemma2:2B, and Llama3.1:8B.

All evaluated LLMs are pre-trained open-source models and are used without additional training or fine-tuning for firewall log data. The proposed framework adopts a retrieval-augmented generation (RAG) approach by incorporating hourly log summaries, event grouping, and database schema information into prompts to support log analysis and SQL query generation.

5. RESULTS

The performance of the proposed decentralized event log management and querying system is evaluated in five aspects: functional correctness, IPFS upload and download latency, blockchain transaction and execution cost, LLM performance and quality, and hourly summarization query efficiency.

5.1 Functional Correctness

The evaluation verifies that all system functions operate correctly in accordance with the architecture framework's components and workflows (Fig. 1). The verification focuses on whether the proposed system can address single-point failures, prevent log tampering, and support auditing through the correct execution of smart contract-based operations.

Fig. 2(a) shows that user registration involves storing the user's blockchain address. After registration, the user is successfully verified and assigned an access role by the administrator through the smart contract.

Fig. 2(b) shows the event log uploading process at the smart contract level. In this step, the Log Manager invokes the `UploadLog` module by submitting the IPFS address of the encrypted event log together with the uploader's EdDSA public key. The IPFS address references the off-chain encrypted log file stored in IPFS. At the same time, the public key is recorded to verify the log source identity and support subsequent integrity and authenticity verification during auditing.

Fig. 2(c) shows the log count process, which returns the total number of log records stored on the blockchain. In the illustrated example, the returned value is 6, indicating that six log entries are currently recorded and available for subsequent selection in the workflow.

Fig. 2(d) illustrates the log retrieval process. Based on the log entry selected in the previous log-counting step (Fig. 2(c), which returns the total number of stored logs), the framework queries the smart contract with logId = 6 to retrieve the corresponding log metadata. The returned tuple includes the IPFS address of the encrypted log file, the uploader's blockchain address, the uploader's EdDSA public key, and the timestamp of log submission.

Fig. 2(e) shows the auditing interface used to verify the authenticity and integrity of a retrieved log file. The interface displays the result of GnuPG signature verification, indicating that the log file is encrypted using Curve25519 and digitally signed with an EdDSA (Ed25519) key. The output confirms a valid signature from the log uploader's identity and shows the signature's creation time, which matches the recorded timestamp. A "Good signature" result indicates that the log file has not been modified after signing, supporting source verification and auditability.

Fig. 2(f) shows the web-based user interface for querying the LLM of the proposed system. The interface allows users to submit natural-language search queries to retrieve and analyze event logs. In the illustrated example, the user submits "which IP address had the most failed login attempts today?"

The DuckDB results show the IP addresses 192.168.1.20 and 192.168.1.21, with 2 and 1 failed login attempts, respectively. The LLM summary indicates that 192.168.1.20 has the most failed login attempts recorded today, with

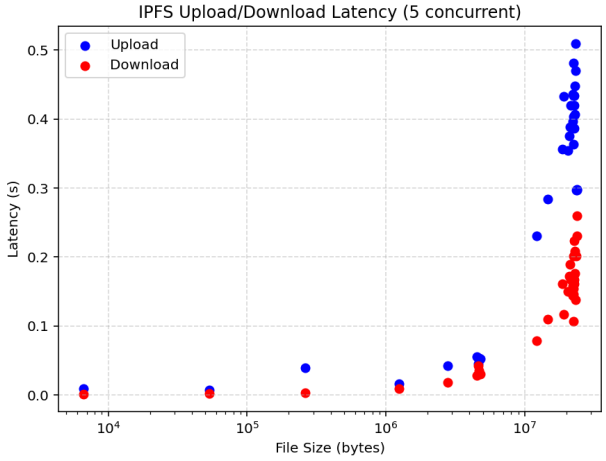

***Fig. 3**: IPFS Upload and Download Latency (5 concurrent).*

2 occurrences. The short response is due to predefined summarization constraints that present only the most relevant information for rapid inspection.

In conclusion, the experimental results shown in Fig. 2(a)–(f) demonstrate that all system functions operate correctly throughout the entire workflow, from log collection and storage to LLM-based querying, without errors. The use of blockchain and IPFS enables decentralized log management and immutable metadata recording, addressing single-point-of-failure and log-tampering issues. In addition, the auditing functionality illustrated in Fig. 2(e) confirms that stored log metadata can be reliably verified.

5.2 IPFS Upload and Download Latency

The performance of the decentralized storage system is evaluated by measuring upload and download times for log files ranging from 50 KB to 50 MB, which represent typical log sizes in real-case deployments. The experiments are conducted on the IPFS network using five concurrent upload and download tasks to emulate multi-user access under moderate workload conditions.

Latency was measured as the time required to upload and download log files through IPFS. The results are presented in Fig. 3. The average upload and download times were 0.41 s and 0.19 s, respectively. Processing time increased with file size. Files larger than 1 MB (10^6 bytes) generally required more processing time, whereas files larger than 10 MB (10^7 bytes) showed greater variation in processing time.

Upload operations take longer than download operations because IPFS splits and distributes files across multiple nodes during storage, whereas downloads typically retrieve data from nodes where the content is cached. The latency results indicate that IPFS can handle large log files with acceptable upload and download delays. The observed latency remains acceptable for large log files, as prior studies show that IPFS achieves upload and download times of only a few seconds even for 100 MB files, significantly outperforming on-chain storage [9].

***Table 4**: Transaction and Execution Cost of Smart Contract Functions.*

Function	Transaction Cost (gas)	Execution Cost (gas)
Upload Log Metadata	666,928	633,936
Get Log Record	-	63,326
Add Security Audit	25,732	4,300

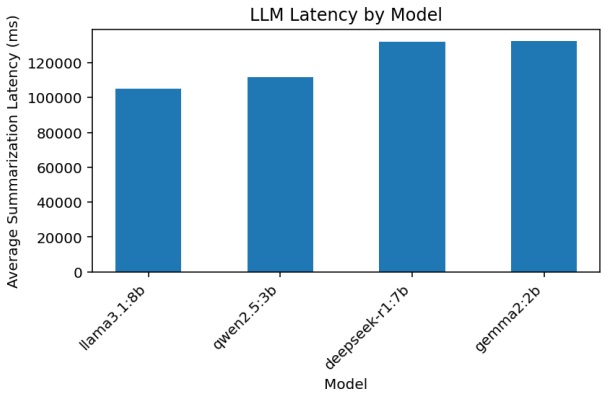

***Fig. 4**: Average Latency of LLM Summarization by Model.*

5.3 Blockchain Transaction Cost

A smart contract is implemented in Solidity to record log file metadata, and the Remix IDE is used for deployment and gas cost evaluation. The experimental results in Table 4 show that the `UploadLogMetadata()` function requires 666,928 gas for transaction execution, corresponding to approximately 0.052 USD, and 633,936 gas for execution, equivalent to 0.049 USD, calculated using a gas price of 0.026 gwei [22] and an ETH price of 2,982.07 USD [23]. The `GetLogRecord()` function incurs no transaction cost because it is a read-only operation. The estimated execution cost is 63,326 gas, which corresponds to approximately 0.005 USD. The `AddSecurityAudit()` function, which registers a security auditor in the smart contract, requires 25,732 gas (approximately 0.002 USD) for transaction execution and 4,300 gas (approximately 0.0003 USD) for execution.

5.4 Query Execution and LLM-Based Summarization

Query execution and result summarization are evaluated using a set of 200 questions, based on three main metrics: response latency, summary length, and LLM-as-a-Judge answer quality score. The models' evaluation is conducted using Llama 3.1:8B, Qwen2.5:3B, DeepSeek R1:7B, and Gemma2:2B, which are referred to as Llama, Qwen, DeepSeek, and Gemma, respectively.

5.4.1 LLM Model Execution Latency

The execution latency of each LLM model is evaluated to compare the response generation time. The experimental results are shown in Fig. 4.

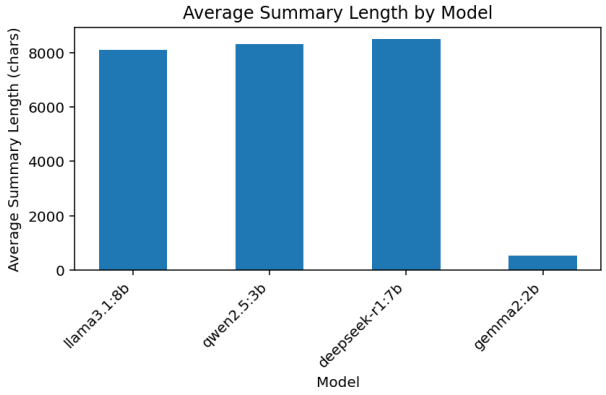

***Fig. 5**: Average Summary Length by Model.*

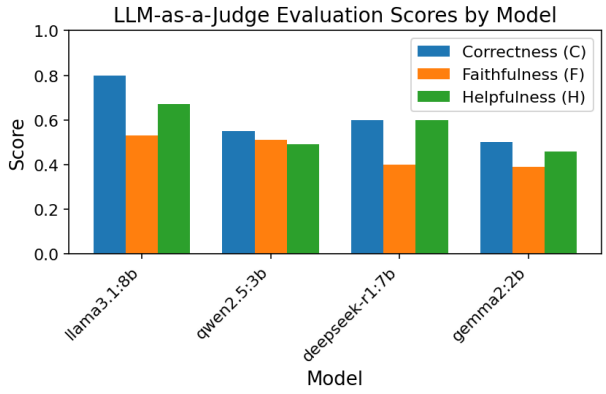

***Fig. 6**: LLM-as-a-Judge Evaluation Scores by Model.*

Llama achieved the lowest average response time at 104,982.69 ms per query, followed by Qwen at 111,517.13 ms. DeepSeek and Gemma exhibited higher latency, with average response times of 132,105.77 ms and 132,469.02 ms, respectively. These results indicate that Llama provides the lowest response latency.

5.4.2 Generated Response Length

The generated response length, used to measure the amount of text produced by each LLM, is evaluated to compare response lengths across models for the same set of questions. Fig. 5 shows that DeepSeek generates the longest responses, with an average length of 8,512.93 characters, followed by Qwen at 8,318.45 characters and Llama at 8,112.42 characters. In contrast, Gemma produces much shorter responses, averaging 524.56 characters. These results indicate that Gemma tends to generate more concise answers, which may lack crucial contextual information.

5.4.3 Content Quality Assessment

The quality of the answers is evaluated using the LLM as a Judge method. Fig. 6 presents the average scores for each model across three aspects: correctness, faithfulness to real data, and usefulness to users.

The results show that Llama achieved the highest scores across all dimensions, with correctness, faithfulness, and helpfulness scores of 0.80, 0.53, and 0.67, respectively, indicating that this model produces more accurate, faithful, and helpful answers. DeepSeek R1 (7B) and Qwen 2.5 (3B) obtained scores of approximately 0.50–0.60, which are lower than those of Llama 3.1 (8B) across all evaluation aspects. Gemma 2 (2B) achieved the lowest scores across all three aspects, consistent with its smaller model size.

5.4.4 Comparative Performance Analysis

The experimental results indicate that Llama provides a strong trade-off between response latency and answer quality. Although the response latency remains high in absolute terms, Llama achieves the lowest latency among the evaluated models and attains the highest scores across all evaluation dimensions, including Correctness (0.80), Faithfulness (0.53), and Helpfulness (0.67). In contrast, DeepSeek produces more detailed answers but requires noticeably longer processing times. Gemma shows similar latency to smaller models but generates shorter responses with lower content accuracy. Based on these findings, the Llama model is selected as the primary model of the framework because it delivers the highest overall performance in terms of data accuracy, consistency with actual results, and usefulness for security event analysis.

5.5 Effect of Hourly Summarization on Query Efficiency

Experiments were conducted using the same DuckDB dataset employed in the proposed framework to evaluate the effect of hourly summarization on query efficiency. The experiment measured the execution time of an aggregate query for counting security event records on both the raw log and hourly summary tables in DuckDB. Query execution was repeated 20 times, and the average query execution time was calculated. The results showed that hourly summaries reduced query time from 0.57 ms to 0.38 ms, representing an improvement of approximately 33.3% compared with raw logs.

This improvement was achieved because hourly summaries reduce the amount of data scanned during query execution by aggregating multiple raw log records into compact summary records. As a result, the system can answer aggregate queries more efficiently before retrieving detailed raw logs when needed.

6. CONCLUSION

This study presents a blockchain-based decentralized log management framework with LLM-assisted querying to improve the security, integrity, and accessibility of security event logs while simplifying the analysis of complex log data. To overcome the limitations of centralized logging and rule-based query approaches, the proposed framework combines blockchain, IPFS, and LLM-based natural-language querying.

The experimental results demonstrate that the proposed system supports secure and reliable log storage and enables efficient analysis of large-scale log data. Llama 3.1 (8B) achieved the highest overall performance, with scores of 0.80 for correctness, 0.53 for faithfulness, and 0.67 for helpfulness. Furthermore, the hourly summarization mechanism improved query efficiency by reducing the average query time from 0.57 to 0.38 ms, corresponding to an improvement of approximately 33.3%. Future work will extend the framework to support multiple security log sources beyond firewall logs, enabling cross-source event analysis for broader security monitoring across organizational and university environments.

REFERENCES

[1] D. Granata, M. Mastroianni, M. Rak, P. Cantiello, and G. Salzillo, "GDPR compliance through standard security controls: An automated approach," *Journal of High Speed Networks*, vol. 30, no. 2, pp. 147–174, 2024.

[2] J. L. Salas-Riega, Y. Riega-Virú, M. Ninaquispe-Soto, and J. M. Salas-Riega, "Cybersecurity and the NIST Framework: A Systematic Review of its Implementation and Effectiveness Against Cyber Threats," *International Journal of Advanced Computer Science and Applications*, vol. 16, no. 6, pp. 723–735, 2025.

[3] M. Podrecca, G. Culot, G. Nassimbeni, and M. Sartor, "Information security and value creation: The performance implications of ISO/IEC 27001," *Computers in Industry*, vol. 142, Art. no. 103744, 2022.

[4] A. Horváth *et al.*, "Anomaly Detection Algorithms for Real-Time Log Data Analysis at Scale," *IEEE Access*, vol. 13, pp. 136288–136311, 2025.

[5] A. I. Swapna, Z. Rahman, M. H. Rahman, and M. Akramuzzaman, "Performance evaluation of fuzzy integrated firewall model for hybrid cloud based on packet utilization," in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, Wuhan, China, 2016, pp. 253–256.

[6] C.-S. Chao, "Firewall Rule Optimization Mechanism for IPv6-Based IoT Networks," in *2024 IEEE 4th International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, Taipei, Taiwan, 2024, pp. 99–104.

[7] W. Li, Y. Feng, N. Liu, Y. Li, X. Fu, and Y. Yu, "A secure and efficient log storage and query framework based on blockchain," *Computer Networks*, vol. 252, Art. no. 110683, 2024.

[8] M. H. Rakib, S. Hossain, M. Jahan, and U. Kabir, "A Blockchain-Enabled Scalable Network Log Management System," *Journal of Computer Science*, vol. 18, no. 6, pp. 496–508, 2022.

[9] S. R. Mallick, R. K. Lenka, and S. Sobhanayak, "Secure and scalable dual blockchain and IPFS-driven IoT ecosystem for next-generation healthcare systems," *Scientific Reports*, vol. 15, Art. no. 41064, 2025.

[10] M. Alsabaan, J. P. Bharadiya, V. Eswarakrishnan, A. M. Cheema, Z. B. Faheem, and J. Ali, "Decentralized Authentication and Secure Distributed File Storage for Healthcare Systems Using Blockchain and IPFS," *Computers, Materials & Continua*, vol. 85, no. 1, pp. 1135–1160, 2025.

[11] P. Kumar, M. Gupta, and R. Kumar, "Improved Cloud Storage System Using IPFS for Decentralised Data Storage," in *2023 International Conference on Data Science and Network Security (ICDSNS)*, Tiptur, India, 2023, pp. 1–6.

[12] S. Deaton, D. Brownfield, L. Kosta, Z. Zhu, and S. J. Matthews, "Real-time regex matching with Apache Spark," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, USA, 2017, pp. 1–6.

[13] P. Jain and A. Mahalkari, "Decentralize Log File Storage and Integrity Preservation using Blockchain," *International Journal of Computer Science and Information Technologies*, vol. 11, no. 2, pp. 21–30, 2020.

[14] M. S. Islam and M. S. Rahman, "LogStamping: A blockchain-based log auditing approach for large-scale systems," *arXiv preprint*, arXiv:2505.17236, 2025.

[15] A. Shahzad, W. Chen, Y. Zhang, and R. Kumar, "Zero-Trust Medical Image Sharing: A Secure and Decentralized Approach Using Blockchain and the IPFS," *Symmetry*, vol. 17, no. 4, Art. no. 551, 2025.

[16] W. Guan, J. Cao, S. Qian, J. Gao, and C. Ouyang, "LogLLM: Log-based Anomaly Detection Using Large Language Models," *arXiv preprint*, arXiv:2411.08561, 2025.

[17] S. Hansakul and N. Wisitpongphan, "Enhancing Business-Specific Phishing Chat Detection via Few-Shot Learning LLM Augmentation," *IEEE Access*, vol. 13, pp. 183589–183597, 2025.

[18] P. S. Rani and S. B. Priya, "Security-Aware and Privacy-Preserving Blockchain Chameleon Hash Functions for Education System," *ECTI-CIT Transactions*, vol. 17, no. 2, pp. 225–234, Jun. 2023.

[19] C. Angsuchotmetee and P. Setthawong, "Block-VOTE: An Architecture of a Blockchain-based Electronic Voting System," *ECTI-CIT Transactions*, vol. 14, no. 2, pp. 174–189, Apr. 2020.

[20] M. S. Rahman *et al.*, "A blockchain-based framework for drug security: Leveraging EdDSA to prevent counterfeiting," *Array*, vol. 28, Art. no. 100604, 2025.

[21] L. Zheng *et al.*, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," *arXiv preprint*, arXiv:2306.05685, 2023.

[22] Etherscan, "Ethereum Gas Tracker," 2025. [Online]. Available: https://etherscan.io/gastracker. [Accessed: Dec. 20, 2025].

[23] CoinMarketCap, "Ethereum (ETH) price," 2025. [Online]. Available: https://coinmarketcap.com/currencies/ethereum/. [Accessed: Dec. 20, 2025].

Warayut Saena received the B.Eng. degree in Computer Engineering from Rajamangala University of Technology Phra Nakhon, Thailand, in 2010, and the M.Eng. degree in Computer Engineering from Mahidol University, Thailand, in 2020. He is currently studying Electrical and Computer Engineering at Khon Kaen University, Thailand. He worked as a Database Administrator and later as a Network Administrator at Phuket Rajabhat University. He is currently a Lecturer in the Industrial Technology Program, Faculty of Science and Technology, Phuket Rajabhat University. His research interests include machine learning, predictive analytics, and optimization algorithms.

Chayada Surawanitkun received the B.Eng. degree in Electrical Engineering (1st class honor) and the Ph.D. degree in Electrical Engineering from Khon Kaen University, Thailand, in 2009 and 2013, respectively. She was granted the B.Eng. Scholarship from the Industrial/University Cooperative Research Center in HDD Component, supported by the Ministry of Science and Technology, and a Ph.D. scholarship from the Royal Golden Jubilee Ph.D. Program under The Thailand Research Fund. She is Program Chair of Master's and Doctoral programs in Electrical and Computer Engineering of the Faculty of Interdisciplinary Studies, Department of Technology and Engineering, Khon Kaen University, Nong Khai Campus, Thailand. Her research interests include power systems, data storage technology, magnetic random-access memory, thermal analysis of nanodevices, nanotechnology, renewable and sustainable energy, and energy management systems.

Manussawee Nokkaew received the Ph.D. degree in Electrical and Computer Engineering from Khon Kaen University, Thailand, in 2024. From 2015 to 2025, he worked as an analyst and consultant at Software Park Thailand under the National Science and Technology Development Agency (NSTDA). During this period, his work focused on managing projects to empower software and digital entrepreneurs in accessing domestic and international markets, and driving digital transformation among industrial enterprises. Currently, he is a Lecturer in the Computer and Information Science Program, Technology and Engineering Sector, at Khon Kaen University, Nong Khai Campus. His research interests include artificial intelligence (AI), particularly generative AI and natural language processing (NLP), as well as machine learning, deep learning, big data analytics, and sustainability.

Kanda Sorn-In received the B.Sc. and M.Sc. degrees in Computer Science in 2002 and 2005, respectively, and the Ph.D. degree in Information Studies in 2014, all from Khon Kaen University, Thailand. She is currently an Assistant Professor in the Computer and Information Science Program, Department of Technology and Engineering, Faculty of Interdisciplinary Studies, Khon Kaen University, Thailand. Her research interests include data analytics, information systems design, artificial intelligence and machine learning, explainable AI, business intelligence, recommender systems, sentiment analysis, and digital innovation for decision support in business, education, and healthcare contexts.

Nongram Mueanrit received her Ph.D. in Information Studies from Khon Kaen University, Thailand, in 2013. She is currently serving as an Assistant Professor in the Department of Technology and Engineering, Faculty of Interdisciplinary Studies, Khon Kaen University. Her research interests encompass a broad spectrum of interdisciplinary domains, including information behavior analytics, consumer behavior analysis, user experience design, business data analytics, and smart tourism systems. She has contributed to the academic community through publications in internationally recognized peer-reviewed journals, including Energy, among others.

Satit Kravenkit received the B.Sc., M.Sc., and Ph.D. degrees in Computer Science from Khon Kaen University, Thailand, in 2001, 2006, and 2017, respectively. From 2004 to 2006, during his graduate studies, he worked as a researcher in semantic web technology with the Department of Computer Science, Khon Kaen University. He later served as a lecturer with the Department of Computer and Information Science, Faculty of Interdisciplinary Studies, Khon Kaen University, Nong Khai Campus. From 2023 to 2026, he worked with the College of Computing at Khon Kaen University. His Ph.D. thesis focused on developing a traceability system compliant with the GS1 EPCglobal Network for cloud-based Electronic Product Code Information Services (EPCIS). His research interests include radio frequency identification (RFID), cloud computing, product traceability and recall in supply chains, blockchain technology, and cybersecurity.