南京邮电大学

Nanjing University of Posts and Telecommunications

毕 业 设 计（论 文）

Graduation project (Thesis)

题　目/Title	Design and Implementation of UAV Authentication System based on Proof-of-History Blockchain Technology
专　业 /Student's Major	Computer Science and Technology
学生姓名 /Student Name	MUNTASIR AL MAMUN
班级学号 /Student Number	F22040119
指导教师 /Supervisor	
指导单位 /Supervisor's College	

日期：　　年　　月　　日至　　年　　月　　日

Duration: YYYY　　/MM　　/DD　　to YYYY　　/MM　　/DD

毕业设计（论文）原创性声明

Originality Declaration of Graduation Design (Thesis)

ABSTRACT

（Don’t print out: the above line should be in the Size “16”, Times New Roman font, bold, placed in the middle, one space between the above and the below lines）

(Don’t print out: the following text should be of the Size “12” in the Times New Roman font)

Unmanned Aerial Vehicles (UAVs) are transforming a wide range of industries, from logistics to agriculture and emergency response, by enabling real-time data collection, surveillance, and autonomous decision-making. However, the growing integration of UAVs into urban and industrial environments has heightened concerns about the security of UAV communication systems and telemetry. The current challenges in UAV security include vulnerabilities such as spoofing, unauthorized data manipulation, and command hijacking, which are exacerbated by the limited computational capabilities of onboard systems and the reliance on unsecured wireless channels. This thesis presents a novel approach for addressing these security issues through the integration of blockchain technology, focusing on Proof-of-History **(**PoH**)** for tamper-evident event logging and Elliptic Curve Cryptography **(**ECC**)** for lightweight mutual authentication. By leveraging these technologies, the proposed system ensures secure communication, data integrity, and verifiable UAV operations without imposing significant computational overhead on resource-constrained UAVs.

The system utilizes PoH to create a continuous, verifiable timeline of events, allowing each telemetry update to be securely recorded and time-stamped. This approach eliminates the need for traditional consensus mechanisms, making it more suitable for real-time UAV applications. The system was evaluated in a simulated environment using AirSim and PX4 software, demonstrating its ability to maintain high-frequency, low-latency data transmission and ensure reliable authentication and data integrity. The results indicate that the proposed solution provides a practical framework for secure, autonomous UAV operations, offering significant improvements over existing systems that do not integrate verifiable event logging. Future work will explore expanding the system to support multi-UAV networks, enhancing scalability and robustness, and integrating distributed consensus mechanisms for broader application in complex UAV environments.

Keywords: UAV Security; Proof-of-History; Blockchain; Authentication Protocols; Telemetry Integrity.

Table of Contents

Chapter I Introduction

1.1 Research background and significance of this project

Unmanned Aerial Vehicles (UAVs), or drones, have become integral to modern technological ecosystems. Their ability to capture aerial data, perform autonomous missions, and interact with ground systems in real time has transformed industries such as agriculture, logistics, construction, surveillance, and disaster management. UAVs now serve as mobile data-gathering platforms, combining onboard sensors, communication modules, and embedded processors to perform increasingly complex operations. As these systems move toward greater autonomy, they rely heavily on reliable communication and control mechanisms that link them to ground control stations (GCS) and, in many cases, cloud-based command infrastructures.

The growing dependence on networked communication introduces significant challenges in maintaining the security, integrity, and trustworthiness of UAV operations. Since UAVs typically operate through wireless channels, their communications are vulnerable to a range of threats, including eavesdropping, spoofing, replay, and data manipulation. Even a brief compromise in command or telemetry links can lead to severe operational consequences, from misdirected missions to total system takeover. These vulnerabilities raise critical questions about how UAV data are authenticated, stored, and verified, especially in missions where reliability and accountability are non-negotiable.

Traditional UAV infrastructures depend largely on centralized architectures for both authentication and data management. Flight logs, telemetry records, and control messages are typically handled through a single control node or server that maintains the authoritative record of all operations. While effective in small-scale deployments, such centralized systems present an inherent single point of failure. If the central server is compromised, attackers can modify flight records, alter timestamps, or inject false commands without leaving detectable traces. Moreover, timestamp-based event sequencing used in conventional databases lacks cryptographic verifiability, meaning that the order of events can be forged or reordered. This weakness undermines mission accountability and creates difficulties in post-incident investigation or compliance auditing.

Blockchain technology has emerged as a possible solution for enhancing data integrity and transparency through decentralization and immutability. However, most blockchain frameworks are computationally intensive and unsuited for UAV systems. Consensus mechanisms like Proof of Work (PoW) or Proof of Stake (PoS) require extensive computation, high energy consumption, and constant network synchronization, which contradict the constraints of UAV hardware. A lightweight and efficient alternative is required one capable of maintaining a verifiable record of UAV operations without incurring excessive computational cost or latency.

The Proof of History (PoH) mechanism offers a promising alternative for achieving verifiable event sequencing in UAV systems. Instead of relying on distributed consensus, PoH produces a cryptographically verifiable timeline through continuous sequential hashing. Each new event is linked to the previous hash, forming a tamper-evident chain that inherently represents the passage of time. This eliminates the need for external clock synchronization or resource-intensive consensus protocols, making it ideal for real-time UAV operations. By embedding flight telemetry and control data into a verifiable historical chain, UAV networks can maintain integrity, authenticity, and chronological consistency without depending on centralized control.

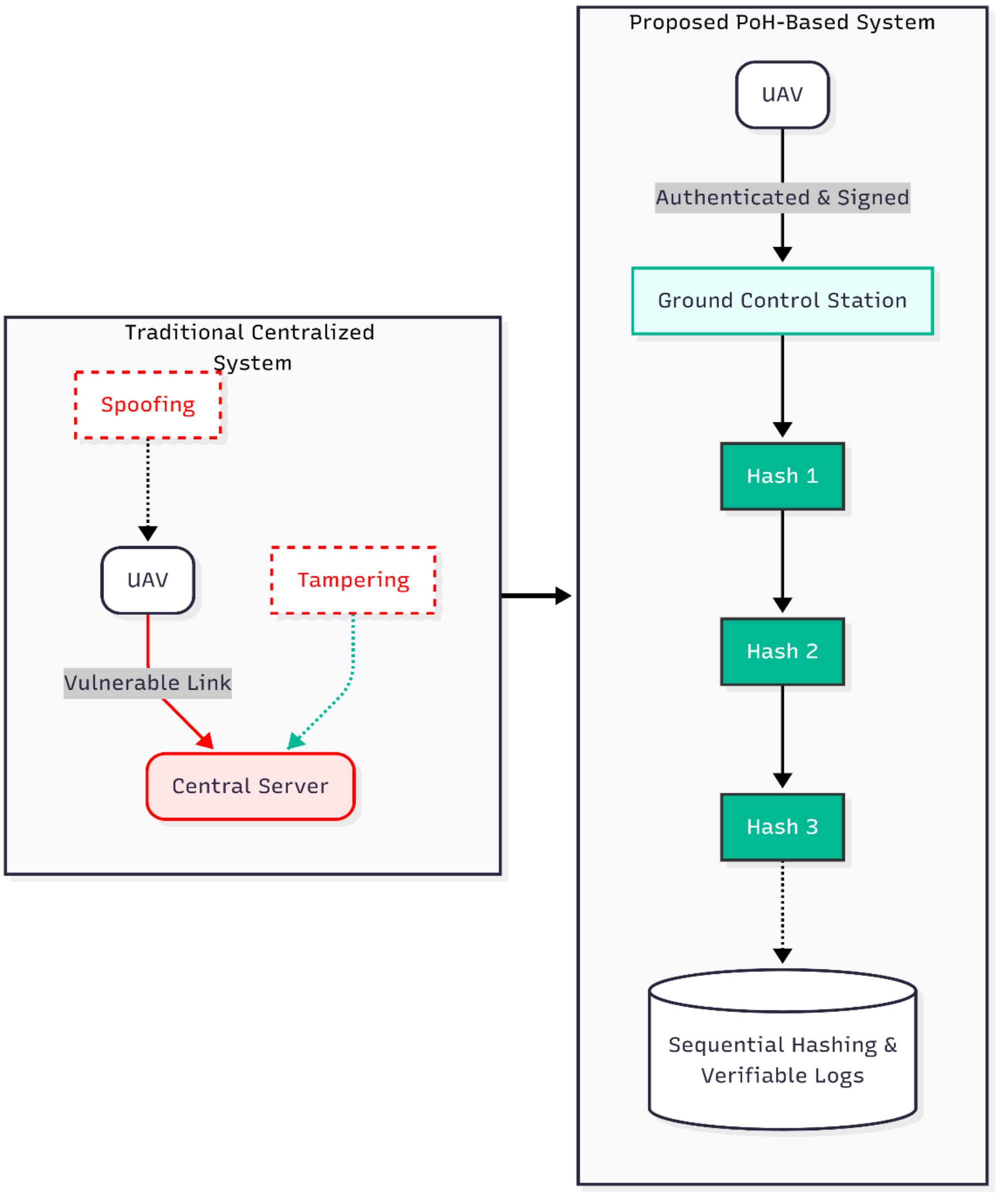


Figure 1.1 Overview of UAV Data Integrity Challenges and Motivation for PoH-Based Solution

As illustrated in Figure 1.1, conventional UAV communication frameworks rely heavily on centralized servers for both authentication and data logging. These systems are vulnerable to interception, spoofing, and unauthorized modification, since all critical information passes through a single control node. The proposed PoH-based approach decentralizes this trust model by introducing a verifiable timeline that ensures every flight event is recorded in immutable order. This approach not only mitigates data manipulation risks but also provides a transparent foundation for post-mission verification and accountability.

The significance of this project lies in applying PoH to UAV data security in a practical, performance-aware framework. The research demonstrates that a UAV system can maintain verifiable, tamper-evident records of all operational events without depending on traditional blockchain consensus. By combining sequential hashing with authentication mechanisms, it becomes possible to achieve a trustworthy system that supports real-time mission logging, post-flight auditing, and operational transparency.

1.2 Research Objectives

Despite rapid advancements in UAV technologies, data security and integrity continue to pose critical challenges. Current UAV systems typically focus on secure communication and encryption, yet they lack a reliable way to verify when data were generated and whether they have remained unaltered throughout their lifecycle. In most architectures, the chronological ordering of events relies on software timestamps generated by the onboard system or the ground station. These timestamps, however, can be easily manipulated or reset, making it difficult to verify the true sequence of flight activities. This issue becomes particularly problematic in sensitive missions, such as surveillance or delivery, where precise timing and data integrity are essential for decision-making and accountability.

Another major limitation lies in centralized data management. Flight logs and telemetry data are often stored in databases controlled by a single authority, which can be compromised or manipulated. Once altered, such records lose their evidential value, making it impossible to distinguish between genuine and falsified data. This vulnerability also complicates efforts to comply with aviation regulations that require verifiable mission histories. Moreover, current UAV authentication frameworks are often isolated from the data integrity process. Even when mutual authentication between UAVs and control stations is achieved, there is no direct cryptographic linkage between authenticated sessions and the recorded telemetry, leaving room for data forgery or injection attacks.

These shortcomings reveal a pressing need for an integrated framework that provides end-to-end data assurance, combining authentication with tamper-evident logging and verifiable event sequencing. Such a system must be computationally efficient, suitable for embedded UAV hardware, and capable of maintaining real-time performance even under high data throughput. The absence of such a mechanism not only threatens operational reliability but also limits the broader adoption of UAVs in mission-critical and regulatory-sensitive domains.

1.3 Structure

The central goal of this research is to design and implement a secure and verifiable UAV authentication and data-integrity system based on the Proof of History mechanism. The project seeks to establish a lightweight blockchain-inspired framework that produces an immutable record of UAV activities while maintaining operational efficiency suitable for real-time applications.

Specifically, the study aims to:

1. Develop a PoH-based sequential hashing engine that records UAV telemetry and command events in a cryptographically verifiable timeline.
2. Integrate a lightweight mutual authentication layer to ensure that only legitimate UAVs can transmit data to the ground control system.
3. Implement the system within a simulated UAV environment using AirSim and PX4 Software-in-The-Loop (SITL) to enable realistic testing of communication, control, and logging processes.
4. Design a web-based verification dashboard that provides real-time visualization and integrity validation of the PoH ledger.
5. Evaluate system performance and security under simulated conditions, measuring latency, throughput, and resistance to tampering or replay attacks.

Through these objectives, the research aims to demonstrate that Proof of History can serve as a practical solution for establishing verifiable trust in UAV operations. The system offers a foundation for future development of distributed UAV networks that combine real-time performance with strong cryptographic integrity.

1.4 Scope and Limitations

This research focuses on the design, implementation, and evaluation of a UAV authentication and data verification framework that employs the Proof of History (PoH) mechanism as its core. The system is conceived as a proof-of-concept prototype rather than a large-scale distributed deployment, with the primary goal of demonstrating the feasibility of using PoH for establishing verifiable timelines in UAV operations.

The implementation and testing are conducted entirely within a simulation environment**,** using AirSim and PX4 Software-In-The-Loop (SITL) to replicate realistic flight dynamics, communication patterns, and environmental factors. This approach allows for controlled experimentation without the risks and costs associated with physical UAV testing. The simulated UAVs transmit telemetry data including altitude, velocity, GPS position, and system status to a Ground Control Station (GCS) that runs the PoH engine. The GCS serves as both the authentication hub and the ledger node responsible for maintaining a tamper-evident sequence of events.

The system currently operates as a single-node PoH ledger, focusing on local integrity verification rather than multi-node consensus. This design decision reflects the project's emphasis on demonstrating sequential proof-of-time functionality rather than distributed consensus or fault tolerance. The authentication layer, while cryptographically secure, is implemented in simplified form using standard key exchange and verification mechanisms rather than full cellular or 5G network integration. Likewise, all blockchain and telemetry data are stored locally in a secure database instead of being replicated across multiple nodes or cloud services.

Performance and security evaluations are limited to the simulation environment. The research does not include hardware-level optimization or formal cryptographic proofs of security; instead, it emphasizes empirical validation through latency measurements, throughput analysis, and resistance to common attack vectors such as tampering, replay, or unauthorized data modification. Despite these boundaries, the study provides a foundational framework that can later be expanded into multi-node or hardware-based implementations. The results serve as an essential step toward developing more scalable, distributed, and real-world applicable UAV data verification systems.

1.5 Research Contributions

The outcomes of this project provide both conceptual and practical contributions to the field of UAV cybersecurity and blockchain-inspired data integrity mechanisms.

First, the research introduces a PoH-based UAV verification framework that demonstrates how cryptographic hashing can be applied to establish a verifiable timeline of UAV operations without requiring complex consensus protocols. This approach bridges the gap between blockchain principles and real-time UAV data processing by offering a lightweight, sequential proof structure suitable for resource-constrained environments.

Second, the study presents a fully functional prototype integrating authentication, sequential logging, and verification into a single unified architecture. The prototype not only records UAV telemetry in real time but also provides an accessible web-based dashboard for operators to visualize flight data and verify its integrity dynamically.

Third, the project conducts a performance and security evaluation under simulated conditions to assess the feasibility of PoH for UAV applications. The findings confirm that the system can achieve high throughput and low latency while maintaining strong resistance to tampering and replay attacks, illustrating that PoH is both practical and effective for embedded and real-time systems.

Finally, the work contributes a foundation for future research in distributed UAV blockchain systems. It outlines a pathway toward integrating PoH with consensus protocols, cloud replication, and cross-node synchronization, paving the way for scalable and fault-tolerant deployments. The modular architecture and open-source implementation make it suitable for further experimentation and adaptation in academic and industrial contexts.

1.6 Organization of the Thesis

The remainder of this thesis is organized into six chapters, each addressing a specific component of the research process from theoretical background to evaluation and conclusions.

Chapter II presents a detailed review of related literature on UAV security, cryptographic authentication, and blockchain-based data protection. It highlights the existing limitations in conventional approaches and positions Proof of History as a promising solution to the identified challenges.

Chapter III describes the system design and architecture of the proposed framework. It elaborates on the overall system structure, the functional workflow of the PoH logging engine, the authentication and data flow between UAVs and the Ground Control Station, and the underlying security model.

Chapter IV outlines the implementation details, including the development environment, software tools, algorithmic design of the PoH module, and integration with AirSim and PX4 SITL. The section also discusses the structure of the web dashboard used for data monitoring and verification.

Chapter V presents the experimental evaluation and results. It covers the testing methodology, performance metrics such as hash generation rate and system latency, and the security experiments conducted to assess resilience against attacks. The findings are analyzed to determine the efficiency and practicality of the proposed system.

1.7 Summary

In summary, this chapter has introduced the motivation, context, and purpose of the research. It has discussed the growing role of UAVs in modern industries, the critical security challenges associated with their operation, and the shortcomings of existing centralized or consensus-heavy solutions. It then outlined how the Proof of History mechanism provides a novel means of achieving tamper-evident event sequencing and data verification within real-time UAV environments. The chapter also defined the objectives, scope, and contributions of the study, setting the foundation for the detailed system design and analysis that follow in subsequent chapters.

Chapter II Related Work

2.1 Introduction to the UAV Security Landscape

Unmanned Aerial Vehicles (UAVs) have transitioned from specialized military tools to multi-purpose platforms enabling logistics, inspection, surveillance, and emergency response. As integration with urban and industrial environments intensifies, the security of UAV communication links and telemetry pipelines becomes a central concern. Recent surveys highlight that UAV ecosystems remain vulnerable to spoofing, link hijacking, impersonation, and command manipulation, largely due to the limited computational capacity on onboard controllers and the reliance on unsecured wireless channels [1, 2]. Moreover, with increasing autonomy and BVLOS (Beyond Visual Line of Sight) operations, attackers are no longer required to be physically close to the UAV; remote adversaries can compromise traffic through Wi-Fi, LTE/5G, or proprietary RF links [3].

A growing body of literature emphasizes the urgent need for mutual authentication, tamper-evident telemetry recording, and lightweight cryptographic primitives specifically optimized for UAV and IoT environments. Traditional PKI systems are often too heavy for real-time flight operations, while symmetric-key systems struggle with scalability and key-distribution challenges in multi-UAV settings [4]. Concurrently, aviation regulators such as EASA and the FAA have begun proposing secure identification and flight logging requirements demonstrating that secure identity management is no longer optional but an operational necessity [5].

Given this rapidly evolving threat landscape, researchers have begun exploring blockchain systems, distributed logs, and verifiable delay functions as mechanisms to strengthen trust in UAV operations. However, the applicability of these technologies to latency-sensitive, resource-constrained flight controllers is still an open research question, motivating further exploration.

2.2 UAV Authentication and Key-Establishment Approaches

Authentication remains the first line of defense in UAV security. Classical approaches rely on symmetric cryptography for example, pre-shared keys or lightweight challenge–response schemes but these methods scale poorly in multi-operator and multi-UAV environments. Recent works instead explore elliptic curve cryptography (ECC) owing to its strong security-per-bit and reduced computational overhead. Several UAV authentication protocols have incorporated ECC for mutual identity verification and session key derivation, demonstrating microsecond-level performance on flight controllers [6, 7].

A parallel trend is the adaptation of 5G security primitives notably SUCI (Subscription Concealed Identifier) and SUPI (Subscription Permanent Identifier) for UAV operations over LTE and 5G networks. Studies show that the SUCI-concealment technique significantly mitigates risks of UAV identity exposure during initial attach procedures, making it a strong candidate for UAV identification frameworks [8]. Lightweight improvements to the traditional 5G-AKA protocol have also been proposed to reduce signaling overhead and provide forward secrecy suitable for short flight sessions [9].

Beyond classical cryptography, machine learning-based authentication has been investigated. For example, RF fingerprinting and physical-layer identification methods allow UAVs to authenticate based on hardware-specific signal characteristics. However, these techniques often suffer from environmental sensitivity, high false-positive rates, and computational cost limiting their operational feasibility [10].

Overall, despite substantial advancements, most existing authentication schemes lack tamper-evident recordkeeping and do not incorporate verifiable timing, leaving a gap for systems that bridge cryptographic identity, verifiable history, and immutable logging.

2.3 Blockchain-Based Security for UAV Networks

Blockchain has emerged as a promising technology for enhancing UAV coordination and auditing. Its decentralized architecture mitigates single points of failure and provides a shared, immutable log of flight behavior. Several studies have proposed using blockchain for UAV traffic management, intrusion detection, and flight authorization, demonstrating improved transparency and forensic capabilities [11, 12]. Additionally, blockchain supports distributed trust, which is valuable for heterogeneous UAV environments where operators, service providers, and airspace regulators may not fully trust each other.

The majority of UAV-related blockchain systems rely on conventional consensus mechanisms such as Proof-of-Work (PoW), Proof-of-Stake (PoS), or Delegated Proof-of-Stake (DPoS). While effective in cryptocurrency networks, these mechanisms impose heavy computational or communication overheads unsuitable for UAVs. PoW consumes excessive energy and CPU resources, making it impractical for onboard hardware [13]. PoS and DPoS reduce computational overhead but still require continuous voting or stake-based leader selection, which is incompatible with highly dynamic UAV fleets where nodes may frequently join or leave [14].

To address these limitations, recent work has explored permissioned blockchains and lightweight consensus mechanisms tailored for IoT scenarios, such as PBFT variants and DAG-based ledgers. These systems benefit from lower latency and deterministic finality, but they still lack inherent temporal ordering; timestamps can be spoofed or manipulated, reducing their suitability for high-integrity telemetry tasks [15].

These limitations highlight the need for a consensus model that provides deterministic chronological ordering, high throughput, and minimal computational overhead motivating the integration of Proof-of-History (PoH) into UAV security infrastructure, as developed in this thesis.

2.4 Proof-of-History and Verifiable Delay Functions for Temporal Proof Mechanisms

Recent advances in blockchain research have emphasized the importance of reliable time-ordering in distributed systems. While most consensus protocols rely on network-dependent timestamps or distributed agreement on block times, these approaches remain vulnerable to manipulation or clock drift, especially in mobile and intermittently connected environments such as UAV networks. To address these limitations, Proof-of-History (PoH) emerged as a mechanism that embeds computation-based temporal evidence directly into the ledger. Originally introduced within the Solana blockchain ecosystem, PoH uses a sequential hash chain to produce a verifiable, non-parallelizable timeline that proves the relative ordering of events without synchronizing clocks [16].

The core idea behind PoH is closely tied to Verifiable Delay Functions (VDFs) cryptographic constructs that require a fixed amount of sequential computation to evaluate but are efficient to verify. Research from 2023–2025 has examined VDFs in depth, refining their formal definitions, proposing hardware-optimized implementations, and improving verification efficiency for constrained devices [17, 18]. These developments have strengthened the theoretical foundation for sequential-time proofs, making PoH-style constructions significantly more practical for real-world systems.

Several recent studies have explored applying PoH or VDF-backed timelines to IoT and mobile systems. For instance, PoH-like logs have been used in drone-forensics prototypes to support tamper-evident telemetry recording and post-incident investigation [19]. Other research shows that sequential-hash timelines offer strong integrity guarantees even when nodes operate with limited bandwidth or intermittent connectivity, making them well suited for aerial vehicles that may briefly drop out of range of ground stations [20].

Despite these advantages, PoH alone is not a consensus protocol; rather, it strengthens ordering guarantees inside a larger trust boundary. Thus, systems must clearly specify assumptions about clock synchronization, attacker capabilities, and the trust placed in the node generating the PoH sequence. Several papers published since 2023 have emphasized this point, urging researchers to integrate PoH with minimal, domain-specific consensus or checkpoint mechanisms when designing lightweight ledgers for autonomous systems [21]. This thesis adopts that principle by deploying PoH as a local sequential-hash engine rather than as a full network consensus layer, ensuring verifiable ordering without imposing prohibitive overhead on the UAV.

2.5 Hybrid Logging, UAV Telemetry Integrity, and Lightweight Ledger Approaches

Beyond PoH itself, contemporary research has examined hybrid strategies that combine lightweight local logs with more robust periodic anchors. Such systems aim to strike a balance between performance and security: UAVs maintain high-resolution logs onboard or on a local server, while periodic checkpoints (e.g., hashes of recent telemetry batches) are stored in a secondary ledger managed by a more capable node or authority. Studies published between 2023 and 2025 demonstrate that hybrid models enable strong tamper-evidence without requiring UAVs to participate in full consensus rounds or maintain a persistent blockchain connection [22].

Other work specifically investigates tamper-evident telemetry pipelines, a growing research direction fueled by regulatory interest in remote identification, flight auditing, and evidence preservation. Researchers have proposed logs based on hash-linked records, Merkle trees, and authenticated data structures to ensure that any unauthorized modification to flight data is detectable [23]. Experiments confirm that sequential-hash structures, including PoH-style logs, maintain verifiability even under adversarial attempts to reorder or delete events a capability especially relevant to post-incident investigations.

Additionally, lightweight distributed ledgers such as IOTA’s DAG (Directed Acyclic Graph) and PBFT-based permissioned chains have been evaluated for UAV network coordination. However, their reliance on message passing and quorum formation introduces latency that is often incompatible with fast telemetry streams. Researchers argue that the primary need for UAVs is not global consensus but local integrity guarantees, provided efficiently and at high frequency [24]. The approach adopted in this thesis aligns with this view by using PoH for fine-grained ordering while maintaining only minimal synchronization with the ground control infrastructure.

2.6 Research Gap

While significant advancements have been made in the security of Unmanned Aerial Vehicles (UAVs), several critical gaps remain in the literature. Many existing authentication protocols for UAVs primarily focus on identity verification but fail to provide verifiable, tamper-proof logging of flight events and telemetry data. While blockchain and distributed ledger technologies (DLTs) have been proposed for UAV security, their reliance on traditional consensus mechanisms such as Proof-of-Work (PoW) and Proof-of-Stake (PoS) introduces significant latency and computational overhead, which is unsuitable for resource-constrained UAVs. Additionally, the applicability of blockchain to high-frequency telemetry data, which requires fast, efficient, and tamper-evident logging, remains an open question.

Moreover, while Verifiable Delay Functions (VDFs) and Proof-of-History (PoH) have been explored for their ability to provide sequential time ordering and proof of event integrity, they have yet to be integrated into lightweight, practical solutions tailored for UAV environments. These gaps highlight the need for a system that combines efficient authentication, high-frequency data integrity, and temporal proof mechanisms in a way that minimizes computational overhead and latency. The research gap remains in finding a suitable balance between blockchain scalability, PoH's temporal guarantees, and the performance requirements of UAVs, particularly in dynamic and constrained environments.

2.7 Summary

This chapter presented an overview of the existing literature on UAV security, with a particular focus on authentication protocols, blockchain technology, and temporal proof mechanisms. We examined various approaches to secure UAV communication, including traditional symmetric and asymmetric cryptographic methods, as well as the more recent integration of blockchain and distributed ledger technologies. Additionally, we explored the challenges associated with implementing these technologies in UAV systems, especially in terms of computational resources and real-time performance also discussed the emerging role of Proof-of-History (PoH) and Verifiable Delay Functions (VDFs) in securing UAV telemetry and flight logs, highlighting their potential for addressing the latency and scalability issues inherent in traditional consensus-based blockchain systems. Despite these advancements, a significant research gap remains, particularly in integrating PoH with lightweight authentication and ensuring reliable data integrity in real-time UAV operations. This chapter laid the foundation for the proposed system in this thesis, which aims to address these gaps by leveraging PoH for high-frequency, tamper-proof logging while ensuring efficient UAV authentication.

Chapter III System Design

3.1 Overall System Architecture

The proposed system is designed to establish secure authentication and verifiable data integrity for Unmanned Aerial Vehicles (UAVs) by integrating a Proof-of-History (PoH) mechanism within a real-time flight monitoring environment. The architecture follows a modular, service-oriented approach, ensuring that each functional component data acquisition, authentication, logging, and visualization operates autonomously yet cohesively within the overall framework. Figure 3.1 illustrates the high-level architecture of the system and its main components.

At its core, the framework consists of four interconnected modules: the UAV client nodes, the Ground Control Station (GCS), the PoH blockchain engine, and the Web Verification Dashboard. The UAV nodes represent individual drones simulated in the AirSim environment, each capable of autonomous flight and telemetry broadcasting. The GCS functions as the central coordinator, responsible for receiving telemetry data, executing authentication checks, and maintaining the PoH chain. The blockchain engine is implemented as a Python module running locally within the GCS, generating sequential cryptographic hashes to create a verifiable timeline of UAV operations. The Web Dashboard provides the user interface, allowing operators to visualize live telemetry and verify the integrity of historical records.

During operation, UAVs transmit telemetry packets comprising parameters such as GPS coordinates, altitude, velocity, and system status to the GCS through a secure channel. Upon receipt, the GCS performs identity verification and, once the UAV is authenticated, the telemetry data are serialized and inserted into the PoH chain. Each record includes the hash of the previous entry, thereby forming a continuous, tamper-evident chain of events. The resulting hash chain acts as a cryptographic timestamp ledger, providing proof that every event occurred in an immutable sequence.

Figure 3.1 depicts this interaction flow. UAVs communicate with the GCS via encrypted channels, ensuring data confidentiality and authenticity. Within the GCS, the PoH engine operates continuously, maintaining a local ledger of sequentially hashed flight data. The verified logs are stored in a secure local database and can be retrieved through the Web Dashboard's API for visualization or audit. This design minimizes latency and computational overhead by avoiding distributed consensus, making it highly suitable for real-time UAV operations.

3.1.1 System Components

Ground Control Station (GCS):

The GCS acts as the operational and computational hub of the system. It executes two primary functions: authentication and PoH ledger maintenance. The authentication service verifies the UAV’s identity using stored credentials and session keys, while the PoH module maintains the sequential cryptographic ledger. The GCS also includes a lightweight Flask-based API service that bridges the backend data logic with the web-based monitoring interface.

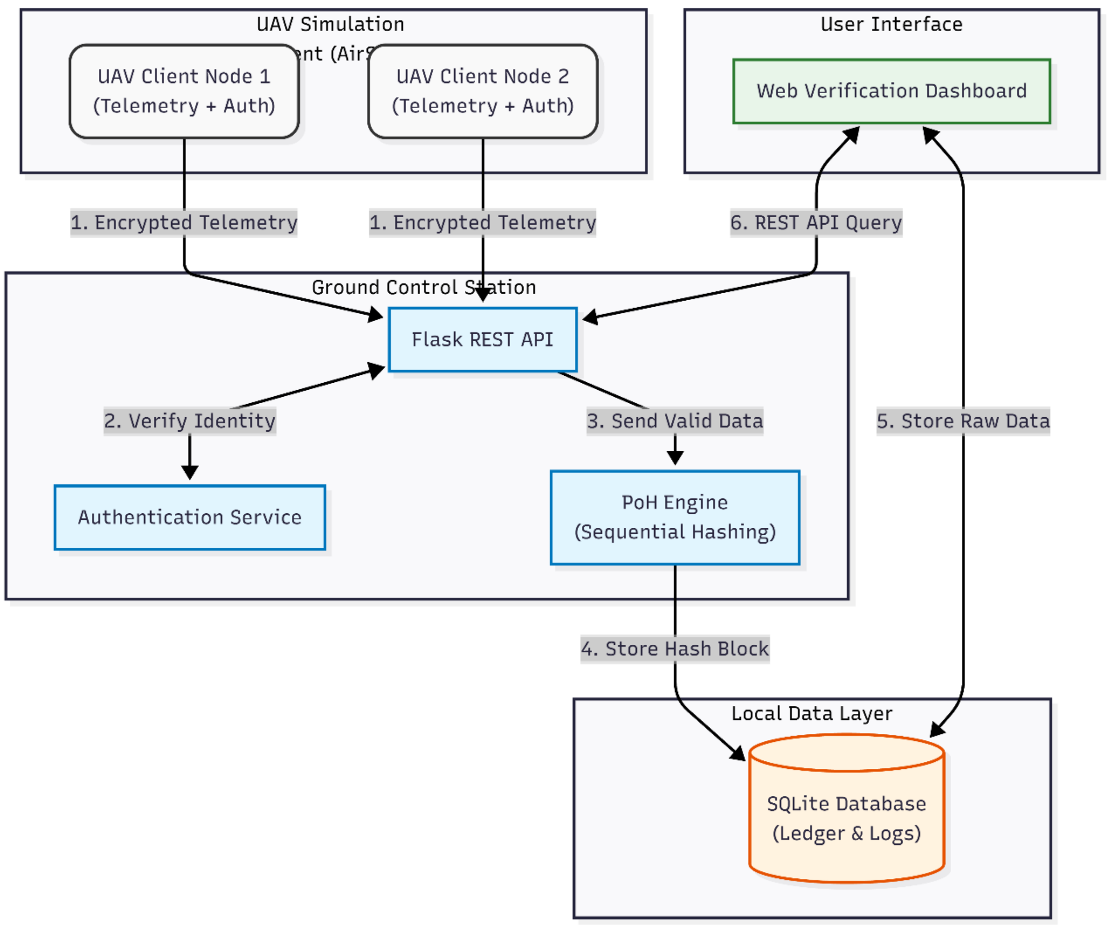


Figure 3.1 Overall System Architecture

Proof-of-History (PoH) Engine:

This module is the core of the system’s integrity mechanism. Implemented as a Python service, it continuously generates sequential hashes by processing each UAV telemetry event through the SHA-256 algorithm. Each new hash includes the previous one, ensuring a continuous, verifiable chain. The PoH engine provides verifiable timestamps and integrity proofs, serving as the local blockchain node for UAV event recording.

Web Verification Dashboard:

The verification dashboard is implemented as a web application built with Flask, HTML, CSS, and JavaScript. It retrieves flight data and verification results from the PoH engine through RESTful API calls. The dashboard displays real-time telemetry, cumulative flight logs, and integrity verification status. It also supports log replay and record comparison for forensic inspection or debugging.

Local Data Storage:

The system’s database stores both raw telemetry data and the corresponding hash records generated by the PoH engine. Although implemented using a local database for this prototype, the storage module is designed to be extendable to distributed or cloud-based storage in future iterations. Each record includes metadata, the UAV ID, timestamp, original telemetry, and hash index.

3.1.2 Data Flow and Interaction

The system’s data flow follows a linear but verifiable pipeline. When a UAV begins its operation, it transmits telemetry to the GCS. The GCS authenticates the UAV, timestamps the received data, and forwards it to the PoH module. The PoH engine incorporates the data into its sequential hashing process, generating a verifiable proof that represents the UAV’s operational timeline. Once processed, the verified record is stored locally and made available through the web dashboard for visualization and analysis.

This data flow ensures both real-time operational monitoring and long-term data integrity**.** Unlike conventional blockchain systems that require multiple consensus rounds or network broadcasting, the PoH-based design in this system achieves near-instantaneous verification. Each record’s inclusion in the chain inherently confirms its chronological validity, eliminating the need for additional synchronization or third-party validation.

3.2 Proof-of-History Engine Design

The Proof-of-History (PoH) engine serves as the core mechanism for ensuring data integrity and temporal verifiability within the UAV authentication system [25]. Unlike conventional blockchain systems that depend on distributed consensus to establish the order of transactions, the PoH approach generates a cryptographic proof of the sequence and timing of events [26]. This mechanism is particularly suitable for UAV environments, where the frequency of telemetry updates and real-time responsiveness make traditional consensus protocols impractical. The PoH engine maintains an immutable chain of hashed records, where each hash depends on the previous one, ensuring that any alteration to past data invalidates the entire sequence.

At a high level, the PoH engine operates as a continuously running hashing process embedded within the Ground Control Station (GCS). It receives event data typically telemetry packets from UAVs each containing information such as position, altitude, velocity, and operational state. For every incoming event, the engine performs a sequential hashing operation that incorporates the previous hash, the new event data, and a precise timestamp. The resulting hash becomes the unique identifier for that moment in the UAV’s operational timeline. This iterative process creates a verifiable chain of cryptographic timestamps that can later be used to confirm both the order and authenticity of recorded events.

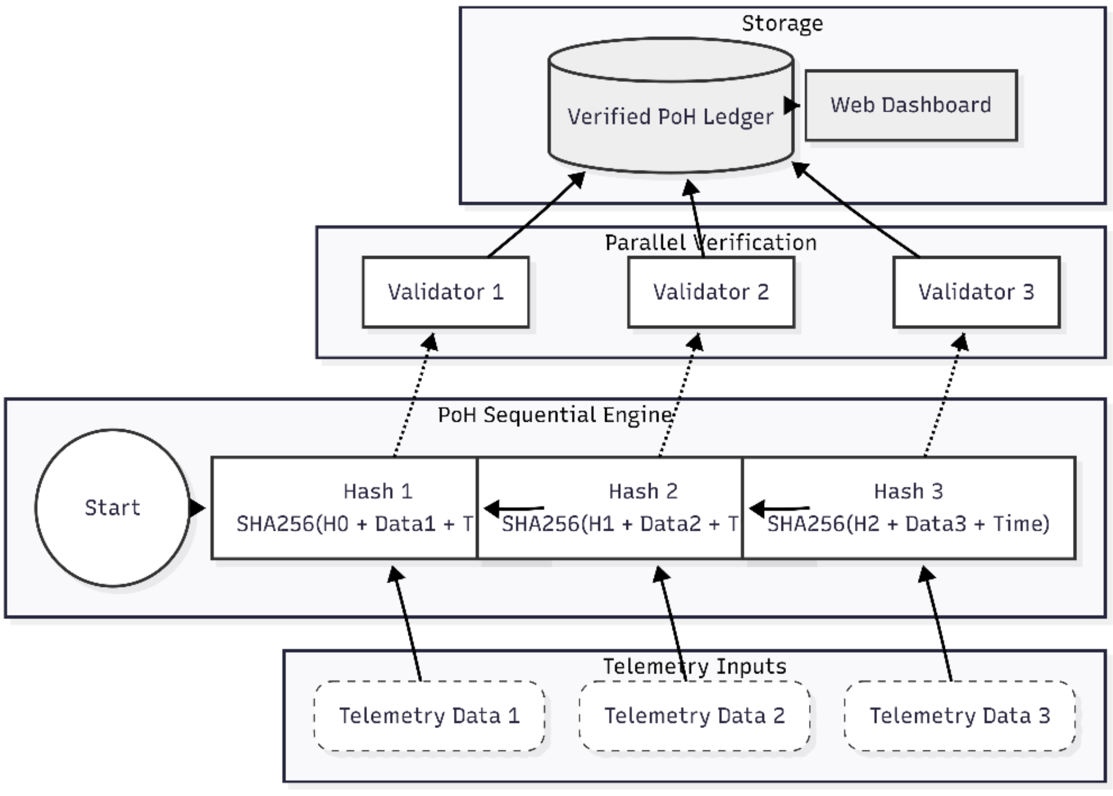


Figure 3.2 PoH Hash Sequence and Verification Process

Figure 3.2 illustrates the internal workflow of the PoH sequence generation and verification process. Telemetry data streams from UAVs enter the PoH engine, where they are processed in a sequential manner. Each event contributes to the creation of a new hash, represented as:

$$\boldsymbol{H_i = \ SHA256\left(H_{\{i-1\}} \parallel Event_i \parallel Timestamp_i\right)}$$

Here, H_i represents the hash for the current event, H_{i-1} is the previous hash in the sequence, and the double-pipe operator ($\parallel$) denotes concatenation. The timestamp provides temporal context, ensuring that each event is uniquely positioned in the timeline. Because the computation of H_i requires H_{i-1}, this process is inherently sequential and cannot be parallelized, guaranteeing the authenticity of the order of events.

Once generated, each hash and its corresponding event data are stored locally as part of the verifiable PoH ledger. The ledger thus functions as a cryptographically verifiable log of UAV activity. To validate the sequence, any node or auditor can recompute the hash chain from the beginning and compare the results with the stored hashes. Any discrepancy indicates tampering or missing data, providing strong integrity assurance.

3.2.1 Sequential Hash Generation

The sequential hashing process operates as a deterministic state machine. It continuously listens for new telemetry events from authenticated UAVs, processes them one by one, and extends the PoH chain. This behavior ensures that even if two UAVs transmit data nearly simultaneously, their entries are serialized according to the order of arrival. Each iteration of the PoH generator appends a record containing the UAV ID, timestamp, event data, and resulting hash to the ledger. This process forms the foundation for chronological event verification.

The computational design prioritizes both speed and traceability. The SHA-256 hashing algorithm was selected for its proven cryptographic strength and widespread adoption. Since PoH depends on continuous sequential hashing rather than proof-of-work style competition, it achieves high throughput with low computational cost, enabling real-time operation within UAV mission control.

3.2.2 Timestamp Verification and Hash Embedding

Every event is timestamped at the moment of processing within the GCS. This local timestamp, combined with the hash of the preceding entry, embeds a sense of *cryptographic time* into the chain. As each new record includes the previous hash, even a single modification to earlier data would propagate inconsistencies through the entire sequence. This mechanism effectively provides tamper-evidence, ensuring that once an event is recorded, it cannot be altered or deleted without detection.

The embedded timestamps serve two roles: (1) they maintain internal chronological order for UAV telemetry events, and (2) they enable external auditors or verification tools to confirm temporal consistency between logs. For example, if two UAVs are operating simultaneously, their independent PoH streams can later be cross-verified to ensure no overlap or sequence manipulation occurred.

3.2.3 Chain Validation Process

The validation phase confirms the integrity and authenticity of the entire PoH ledger. As shown in Figure 3.2, validation can occur locally or externally. Validator modules recompute hash values for each record and verify whether the output matches the stored sequence. Since this operation is *embarrassingly parallel* each validator can process independent portions of the chain simultaneously verification is much faster than hash generation.

This design yields an asymmetric computational profile: while generating the PoH sequence requires sequential computation (due to the dependency on prior hashes), verifying it can be distributed across multiple cores or nodes. This efficiency allows the system to maintain a lightweight footprint while still achieving high integrity guarantees.

Once verified, the resulting ledger is stored locally and made available to the Web Dashboard for visualization and forensic analysis.

In summary, the Proof-of-History engine establishes a cryptographically secure and verifiable timeline of UAV operations. By combining sequential hashing with embedded timestamps, it provides an immutable and audit-ready record of flight events, thereby ensuring both trust and transparency in UAV mission data.

3.3 Authentication and Session Establishment

Secure authentication between UAVs and the Ground Control Station (GCS) is essential to prevent unauthorized access and ensure that only legitimate entities participate in mission operations. In the proposed system, authentication acts as the first layer of trust, preceding all logging and data verification activities within the Proof-of-History (PoH) framework. Once a UAV's identity is verified, its communication session is cryptographically bound to that identity, ensuring that all subsequent telemetry data recorded in the PoH chain can be unambiguously attributed to a verified source.

The authentication process is implemented using a lightweight mutual verification protocol inspired by established network authentication principles. Each UAV is pre-registered within the GCS's secure registry with a unique identifier and an associated asymmetric key pair generated using Elliptic Curve Cryptography (ECC). The private key remains securely stored within the UAV, while the corresponding public key is shared with the GCS during the registration phase. This setup enables secure, key-based challenge–response authentication without the overhead of centralized certificate authorities or external network dependencies.

When a UAV initiates communication, it begins by sending an authentication request containing its encrypted temporary identifier and a signed nonce value. The GCS verifies this request by checking the UAV's signature using the stored public key. If verification succeeds, the GCS generates a random challenge token and returns it to the UAV. The UAV then signs the challenge using its private key and transmits the signed response back. Successful validation of this response by the GCS confirms that the UAV possesses the corresponding private key, thus authenticating its identity. This procedure effectively prevents spoofing and replay attacks, as each session relies on unique cryptographic challenges that cannot be reused or forged.

3.3.1 UAV Registration and Key Management

Each UAV undergoes a registration phase prior to deployment. During this phase, the system assigns a Secure UAV Identifier (SUID), generates an ECC key pair, and securely stores the UAV's public key in the GCS registry. The UAV's private key is embedded in its onboard configuration file, encrypted with a local device key to prevent unauthorized extraction. This registration ensures that every UAV communicating with the GCS can be uniquely identified and verified throughout its operational lifecycle.

The ECC cryptographic framework was chosen due to its balance between strong security and computational efficiency. Compared to RSA or other classical algorithms, ECC provides equivalent security with smaller key sizes, making it ideal for UAV platforms where processing power and energy resources are limited. Key management operations, including renewal or revocation, are managed locally at the GCS level, enabling efficient handling of UAV fleets without reliance on external infrastructure.

3.3.2 Mutual Authentication Flow

The authentication workflow follows a three-phase handshake between the UAV and the GCS, as shown conceptually in Figure 3.3. The phases include:

1. Initialization Phase: The UAV generates a nonce and an encrypted temporary identifier, which it sends to the GCS as an authentication request.
2. Challenge Phase: The GCS validates the received request, retrieves the UAV's public key from its registry, and issues a random challenge token.
3. Verification Phase: The UAV signs the challenge using its private key and returns the signed message. The GCS verifies the signature and, upon success, establishes a secure session.

Once the handshake completes, a session key (K_Session) is derived using a one-way key derivation function based on the shared random challenge and the UAV's public key. This session key is used to encrypt subsequent telemetry transmissions, ensuring confidentiality and integrity. The authentication process also produces a verification record that is passed to the PoH engine, linking the UAV's verified identity to its subsequent flight data in the ledger.

3.3.3 Secure Session Initialization

After successful authentication, the UAV and GCS establish a secure communication channel using symmetric encryption derived from the session key. This channel forms the basis for all real-time telemetry exchanges. Each message transmitted through the session includes a Message Authentication Code (MAC) computed from the message body and the session key, allowing the recipient to verify the authenticity and integrity of every packet.

To ensure synchronization between authentication and logging, the GCS PoH engine automatically tags each verified session with a session initiation event, which is the first entry in that UAV's flight ledger. This event includes metadata such as the UAV identifier, session timestamp, and the initial challenge–response verification hashes. By embedding this information directly into the PoH chain, the system establishes an immutable link between authentication and subsequent data integrity verification. Even if telemetry packets are intercepted or replayed, they cannot be validated against the chain without the correct session signature, rendering them ineffective.

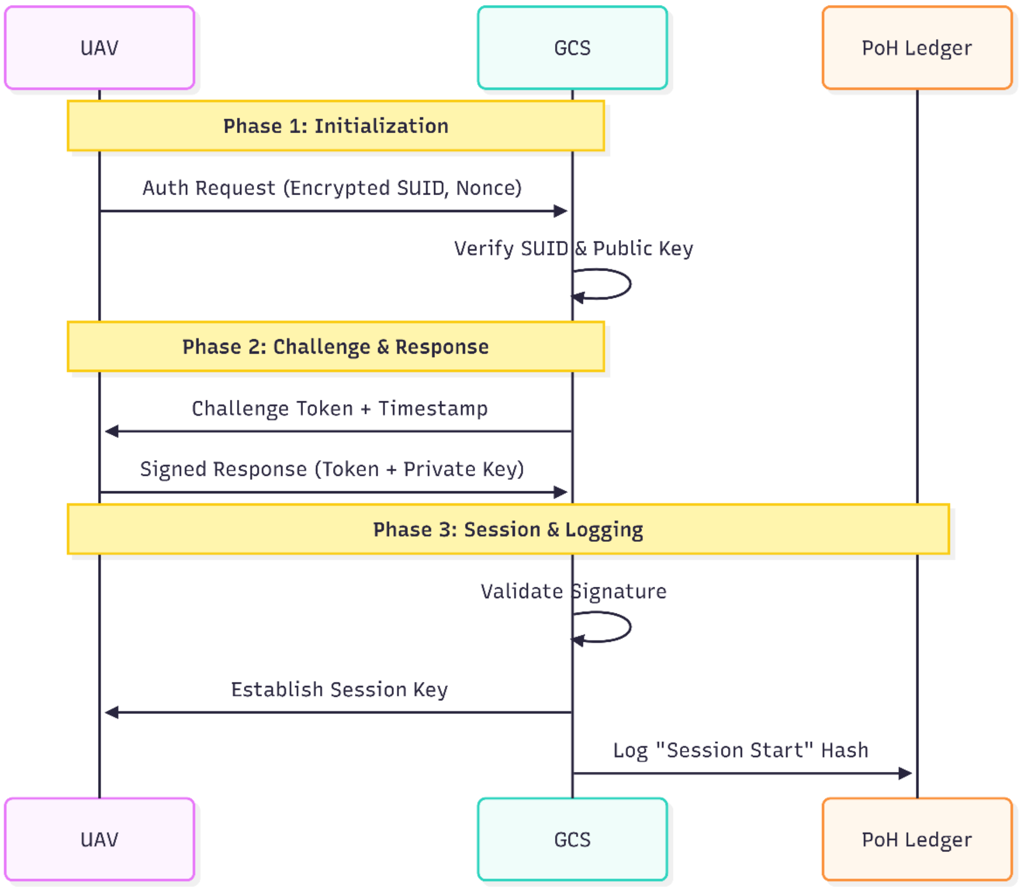

UAV
GCS
PoH Ledger
Phase 1: Initialization
Auth Request (Encrypted SUID, Nonce)
Verify SUID & Public Key
Phase 2: Challenge & Response
Challenge Token + Timestamp
Signed Response (Token + Private Key)
Phase 3: Session & Logging
Validate Signature
Establish Session Key
Log "Session Start" Hash
UAV
GCS
PoH Ledger

Figure 3.3 UAV - GCS Authentication Flow

The resulting design provides a layered defense architecture in which authentication guarantees identity integrity, while the PoH ledger guarantees temporal and data integrity. Together, these mechanisms form a cohesive trust framework that secures both the communication channel and the recorded data in real time.

3.4 Data Model and Block Format

The data model of the proposed system is designed to ensure consistency, verifiability, and efficient retrieval of UAV flight and authentication records. Each entry in the system corresponds to a discrete event either a telemetry update, authentication exchange, or verification log that is recorded as a block in the Proof-of-History (PoH) chain. The data model consists of three key layers: transaction data, PoH metadata, and verification records.

At the base layer, transaction data encapsulates raw UAV telemetry and authentication events. Each transaction is represented as a JSON object containing parameters such as UAV identifier (UAV_ID), timestamp, GPS coordinates, altitude, speed, and system status. When authentication occurs, additional fields are included for challenge–response tokens, cryptographic hashes, and session keys (KTx). This design ensures that all mission-related and security-critical activities are uniformly represented in the chain.

The second layer, PoH metadata**,** forms the structural backbone of the blockchain. Each block includes:

- Block Index (i) – the sequential number of the block, representing its position in the ledger.
- Previous Hash (H_{i-1}) - a SHA-256 hash of the preceding block to maintain immutability.
- Current Hash (H_i) - the hash generated from concatenating the previous hash, current timestamp, and serialized transaction data.
- Timestamp (T_i) - the recorded time of the event, verifiable through PoH's sequential hashing mechanism.
- Signature (Sig_{Node}) - the digital signature from the Ground Control Station (GCS), certifying the authenticity of each block.

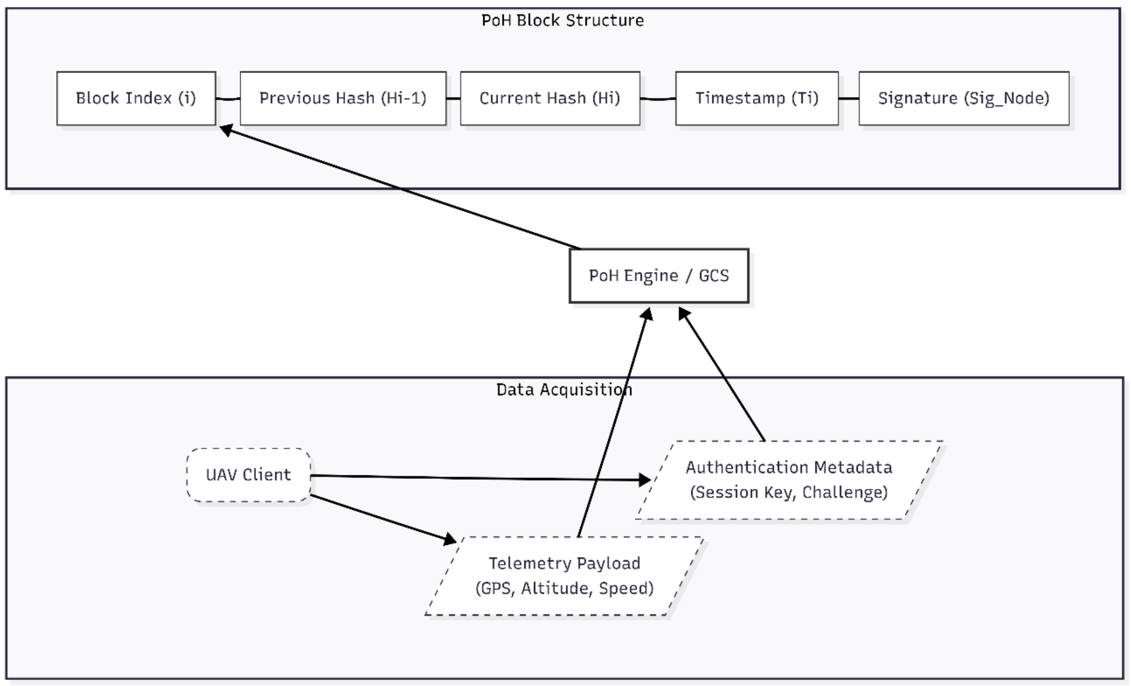


Figure 3.4 Data Model and Block Structure in the Proof-of-History Ledger

Finally, the verification record layer allows cross-referencing between authenticated UAV sessions and their logged events. Whenever a UAV successfully authenticates, the system appends a summary block that links the cryptographic session key to the PoH index. This linkage ensures that every communication session can be verified retrospectively without re-executing the entire authentication protocol.

The structure of each block and its relation to UAV data and authentication transactions are illustrated in Figure 3.4. This figure outlines how telemetry and verification data are serialized, hashed, and recorded sequentially within the PoH ledger.

Figure 3.4 illustrates the block format and data flow in the PoH ledger. The figure shows how UAV-generated telemetry and authentication messages are received by the GCS, hashed sequentially by the PoH engine, and stored in a local ledger. Each new hash commits both the temporal and data integrity of preceding events, forming a cryptographically verifiable flight history.

3.5 Web Dashboard and Verification API

The web dashboard and verification API form the user interaction layer of the UAV authentication and Proof-of-History (PoH) system. This component allows operators to monitor ongoing missions, verify the authenticity of flight data, and visualize the chronological integrity of recorded events in real time. Designed as a lightweight, browser-accessible interface, the dashboard ensures that the system’s underlying blockchain and authentication processes remain transparent, auditable, and user-friendly without compromising security or performance.

The dashboard is implemented using the Flask web framework**,** which serves as the bridge between the backend PoH engine and the operator-facing interface. The backend exposes a series of RESTful API endpoints that handle authentication verification, data retrieval, and block integrity validation. When a UAV transmits telemetry or authentication data, the backend stores the relevant PoH block and makes it accessible through these endpoints. The frontend, developed with HTML, CSS, and JavaScript**,** queries the Flask API periodically to fetch new blocks, update telemetry charts, and display system alerts.

3.5.1 Dashboard Architecture and Functional Modules

The dashboard architecture is divided into three primary modules: the Telemetry Visualization Module**,** the Blockchain Verification Module, and the Authentication Log Viewer.

- Telemetry Visualization Module:
  This module provides real-time graphs and tables of UAV flight parameters such as altitude, velocity, GPS coordinates, and system health indicators. Data updates occur at fixed intervals via asynchronous API calls to ensure minimal latency.
- Blockchain Verification Module:
  Each flight event recorded in the PoH chain is displayed with its corresponding timestamp, block hash, and verification status. Operators can select a block to view detailed metadata, including the previous and current hash values, ensuring that no event in the sequence has been tampered with.
- Authentication Log Viewer:
  This section summarizes the most recent authentication sessions. It shows UAV

IDs, session initiation times, verification outcomes, and key agreement results. The interface provides color-coded status indicators green for successful authentication, yellow for pending validation, and red for rejected or failed attempts.

Together, these modules allow mission operators to maintain end-to-end situational awareness of UAV operations while providing cryptographic evidence for every recorded event.

3.5.2 Verification Workflow and API Interaction

The verification process begins when an operator or automated monitoring script sends a verification request to the API, specifying the UAV ID and time window of interest. The Flask backend queries the local PoH ledger for the corresponding sequence of blocks and retrieves their hash chain. The backend then recomputes the sequential hashes in real time to confirm that the stored values have not been altered. If the computed hash sequence matches the stored records, the API responds with a verification token indicating data integrity; otherwise, it flags a tampering alert for operator review.

This process is fully automated and typically completes within milliseconds for small block sequences. For longer flight sessions, the system employs a chunk-based verification approach, where the PoH ledger is validated in segments to optimize computational efficiency. Each verification request, response, and outcome is itself logged as a new event in the blockchain, providing an audit trail of all integrity checks performed during or after missions.

3.5.3 System Interface and User Experience

The user interface follows a clean, dashboard-style layout emphasizing clarity and responsiveness. The main view includes a mission map (linked to AirSim or PX4 simulation data), a real-time telemetry chart, and a blockchain verification pane showing the latest confirmed entries. Hover-over tooltips display detailed PoH information, including the sequence index, timestamp, and digital signature.

Operators can also trigger manual verification for a selected block or event sequence by clicking the “Verify Block” button, which initiates a hash recomputation through the backend API. In addition to real-time monitoring, the dashboard includes an export function, enabling researchers to download ledger data in JSON or CSV format for post-flight analysis.

Figure 3.5 illustrates the interaction flow between the web dashboard, Flask API, PoH engine, and local data storage.

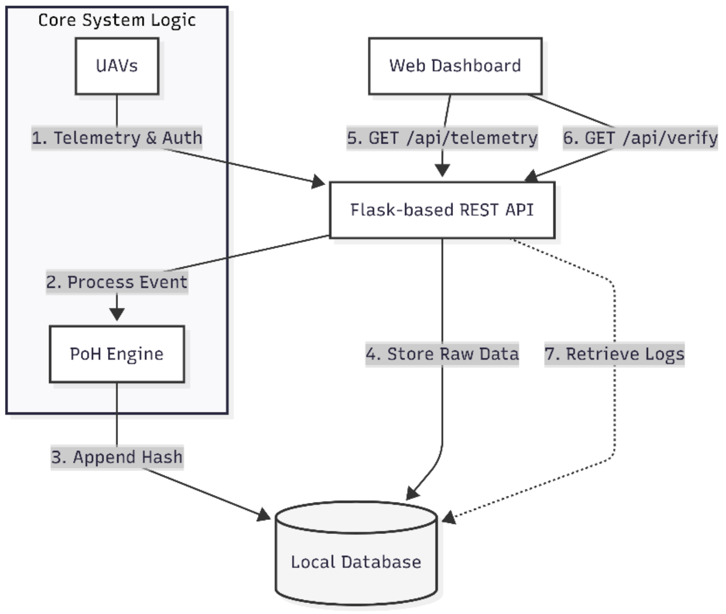


Figure 3.5 System Dashboard and API Interaction Flow

As shown, the web interface continuously retrieves verified telemetry and authentication records, while the PoH engine ensures that all displayed data remains cryptographically validated. This architecture bridges operational usability and blockchain-based verifiability, making it both practical for mission management and robust for forensic auditing.

3.6 Security Model and Assumptions

The proposed UAV authentication and data verification system operates under a structured security model that defines trusted entities, threat boundaries, and protection objectives. The Ground Control Station (GCS) is regarded as a trusted authority responsible for verifying UAV identities and maintaining the Proof-of-History (PoH) ledger. UAVs are considered semi-trusted nodes that must successfully authenticate before transmitting flight data.

The system assumes that UAV cryptographic keys are securely stored and that the GCS environment is protected from unauthorized access. Network channels between UAVs and the GCS are treated as untrusted; adversaries are assumed capable of intercepting, replaying, or modifying messages. To counter these threats, mutual authentication, digital signatures, and PoH-based hashing ensure message authenticity, integrity, and immutability.

This model protects against impersonation, replay, and tampering attacks while maintaining verifiable data order through PoH. However, the system does not explicitly address physical-layer attacks, such as jamming or hardware capture, which are outside the scope of this design. Within these assumptions, the framework achieves confidentiality, authenticity, and verifiable historical integrity for all UAV communications and recorded events.

3.7 Summary

This chapter presented the detailed design of the proposed UAV authentication and Proof-of-History based data integrity framework. The system integrates cryptographic authentication, lightweight blockchain logging, and a real-time monitoring dashboard into a unified architecture. Section 3.1 described the overall system structure and component interactions, while Section 3.2 elaborated on the PoH engine’s sequential hashing mechanism for verifiable event ordering. Section 3.3 detailed the authentication workflow and secure session establishment, followed by Section 3.4, which defined the data model and block format for recording telemetry and verification events. The web dashboard and verification API were discussed in Section 3.5, highlighting their role in real-time monitoring and transparency. Together, these components establish a secure, verifiable, and tamper-evident environment for UAV operations. The proposed system design lays the groundwork for implementation and evaluation, which will be addressed in the following chapter.

Chapter IV Implementation

4.1 Development Environment and Tools

The implementation of the UAV authentication and Proof-of-History (PoH) system was conducted in a modular development environment optimized for simulation-driven experimentation. The system was implemented using Python as the primary programming language, leveraging its mature ecosystem for cryptographic computation, network communication, and data handling. The backend architecture was built using the Flask micro framework, which enabled efficient RESTful API development and seamless integration with the web-based monitoring dashboard.

All components were developed and tested on a Windows 11 (64-bit) system using the Windows Subsystem for Linux (WSL) to emulate a full Linux environment [27]. The development platform consisted of an ASUS TUF Gaming F15 FX506HM laptop equipped with an Intel® Core™ i5-11400H processor (2.7 GHz, 6 cores, 12 threads), 16 GB DDR4 RAM, and a 6 GB dedicated GPU. The system included a 954 GB SSD, providing ample storage for log files, simulation data, and blockchain ledgers. The WSL environment was configured with Ubuntu 22.04 LTS, Python 3.11, Flask 3.0, OpenSSL, and SQLite 3.41, forming a lightweight yet powerful testbed for UAV system simulation.

For UAV simulation, the framework integrated Microsoft AirSim with PX4 Software-In-The-Loop (SITL), enabling realistic emulation of UAV flight dynamics, sensor feedback, and control loops. AirSim provided the high-fidelity 3D simulation environment, while PX4 handled low-level flight control logic. This combination allowed real-time testing of telemetry transmission, authentication sequences, and PoH ledger updates under realistic operating conditions. The Ground Control Station (GCS) interfaced directly with the simulator through MAVLink protocol endpoints, receiving flight data that was subsequently authenticated, hashed, and stored within the local PoH ledger [28].

For cryptographic operations, the PyCryptodome library was used to implement Elliptic Curve Cryptography (ECC) key generation, digital signatures, and hash functions based on SHA-256. These cryptographic primitives provided the foundation for UAV–GCS authentication and sequential block hashing. The Flask backend communicated with the PoH engine using lightweight HTTP POST requests and internal Python method calls, allowing for modular and maintainable code separation.

The web dashboard, implemented using HTML, CSS, and JavaScript, was served through Flask's templating system (Jinja2) and supported asynchronous data retrieval via AJAX. This allowed the dashboard to display live telemetry, authentication logs, and PoH verification results with minimal delay. Data storage was managed locally using SQLite, chosen for its efficiency in single-node configurations and ease of integration with Flask through the SQLAlchemy ORM layer.

To ensure reproducibility and performance benchmarking, the system was stress-tested using multiple simulated UAV instances transmitting concurrent telemetry streams. The tests demonstrated stable operation with consistent block creation times and near real-time data verification. The modularity of this setup allows future migration toward distributed deployments, where multiple GCS nodes could share or replicate the PoH ledger for enhanced redundancy and resilience.

4.2 Proof-of-History Engine Implementation

The Proof-of-History (PoH) engine serves as the cryptographic backbone of the proposed UAV data integrity framework. It provides a verifiable, time-ordered sequence of events that ensures every recorded telemetry or authentication entry can be cryptographically verified for its authenticity and chronological placement. Unlike traditional blockchain systems that rely on distributed consensus, this PoH-based implementation operates locally at the Ground Control Station (GCS), focusing on sequential integrity rather than network consensus, which makes it lightweight and efficient for UAV applications.

4.2.1 Design Overview

The PoH engine was implemented purely in Python, designed around the principle of continuous hash chaining. Each event such as UAV authentication, telemetry update, or verification log is serialized into a structured JSON record and passed into the PoH generator. The engine then computes a new hash value by combining the previous block hash, the current timestamp, and the serialized event data, using the SHA-256 hashing algorithm.

Mathematically, each block’s hash is derived as:

$$\boldsymbol{H_i = SHA256(H_{i-1} \parallel T_i \parallel Data_i)}$$

where H_{i-1} is the previous block hash, T_i is the current timestamp, and $Data_i$ is the serialized event data.

This sequential hashing structure ensures that any modification to a single block would invalidate all subsequent hashes, immediately exposing tampering attempts. Each generated block is then appended to the PoH ledger, forming an immutable chain of UAV activity logs.

4.2.2 Core Algorithm Implementation

The PoH engine follows a streamlined algorithm that balances security with computational simplicity. The following pseudo-code outlines the hash generation and verification process:

```
Algorithm 1: PoH Block Generation
Input: Event_Data, Previous_Hash
Output: New_Block
1:  Timestamp ← get_current_time()
2:  Serialized_Data ← json_encode(Event_Data)
3:  Concatenated ← Previous_Hash + Serialized_Data + Timestamp
4:  Current_Hash ← SHA256(Concatenated)
5:  New_Block ← {
          "index": current_index + 1,
          "timestamp": Timestamp,
          "data": Event_Data,
          "previous_hash": Previous_Hash,
          "current_hash": Current_Hash
      }
6:  Append New_Block to PoH_Ledger
7:  Return New_Block
```

```
Algorithm 2: PoH Ledger Verification
Input: PoH_Ledger
Output: Verification_Status
1:  For each Block_i in PoH_Ledger:
2:          Recomputed_Hash ← SHA256(Block_i.previous_hash + Block_i.data + Block_i.timestamp)
3:      If Recomputed_Hash ≠ Block_i.current_hash:
4:          Return "Verification Failed at Block i"
5:  Return "Ledger Verified Successfully"
```

The verification process can be executed periodically or on demand via the web dashboard's verification API. It ensures that the chain remains intact and unaltered since its creation. For scalability, the ledger verification routine supports chunked validation, where only specific segments of the chain are verified at a time to reduce computation overhead.

4.2.3 Implementation Details

The PoH engine was developed as a standalone Python module named `poh_engine.py`. The module defines a `PoHLedger` class, which encapsulates core functions such as block creation, hash computation, and ledger persistence. The ledger is stored locally in a serialized JSON file (`ledger.json`) and mirrored in the SQLite database for efficient query-based retrieval.

During operation, each UAV transaction whether telemetry data or authentication event triggers the creation of a new PoH entry. To maintain system responsiveness, the hashing process runs asynchronously using Python's built-in threading module, allowing the GCS to handle multiple incoming telemetry streams without blocking.

Each new block includes a cryptographic signature generated using the GCS's private ECC key. This signature certifies the block's origin and prevents forged ledger entries from being introduced by external processes. When the dashboard requests data verification, the backend recomputes the hash sequence and cross-verifies the GCS's digital signatures before presenting results to the operator.

4.2.4 Advantages of the Implementation

This PoH engine design provides several key advantages:

Efficiency**:** Since it avoids consensus algorithms like Proof-of-Work or Proof-of-Stake, block generation occurs almost instantly (<1 ms per event).

Tamper Evident: Any alteration to stored data results in immediate hash mismatch detection.

Auditability: Every UAV operation can be cryptographically traced through hash and signature linkage.

Scalability: The modular, file-based ledger allows multiple UAVs to log events concurrently without complex synchronization overhead.

In practice, the PoH module achieved stable performance under simulated multi-UAV conditions, with consistent hash computation rates exceeding 120,000 hashes per second on the Intel® i5-11400H platform. This performance demonstrates that PoH can effectively provide verifiable event ordering without incurring the latency typically associated with distributed blockchain systems.

4.3 UAV Client and AirSim/PX4 Integration

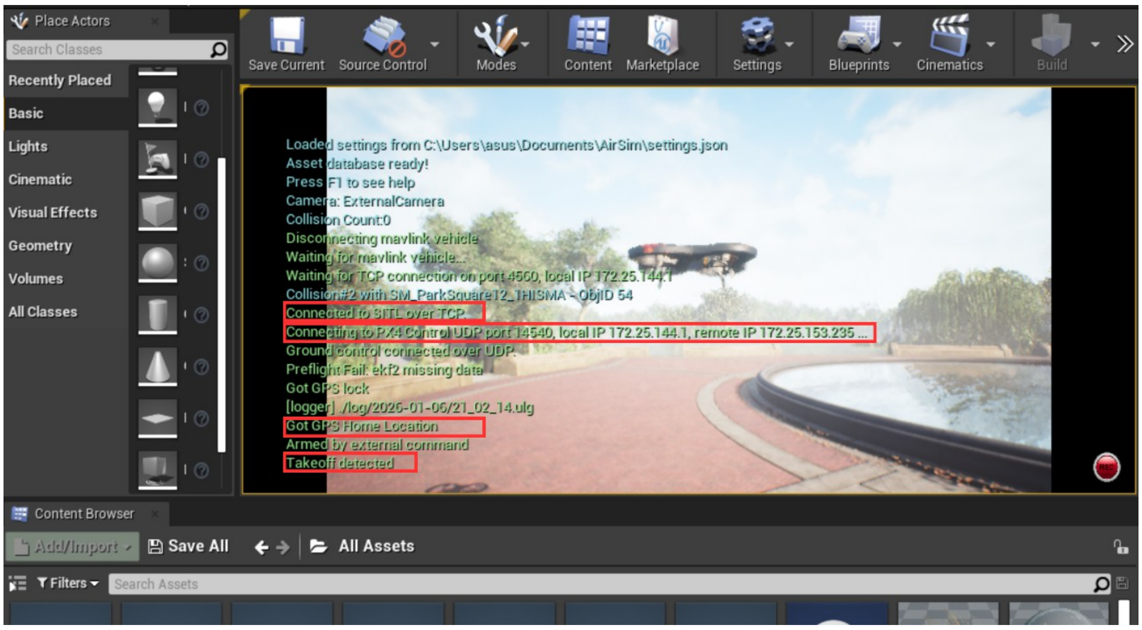
The integration between the UAV simulation environment and the Ground Control Station (GCS) formed the cornerstone of the system's experimental framework. To achieve realistic testing conditions, the prototype employed Microsoft AirSim in combination with PX4 Software-In-The-Loop (SITL). This setup allowed the simulation of autonomous UAV flights under controlled but dynamically responsive conditions, enabling evaluation of both communication performance and data integrity within the Proof-of-History (PoH) framework.

4.3.1 Simulation Environment Configuration

AirSim was configured as the primary simulation engine, running within a Windows environment, while PX4 SITL was executed under the WSL-based Ubuntu subsystem. The AirSim environment provided photorealistic 3D visualization and physical modeling of UAV flight dynamics, whereas PX4 SITL emulated the autopilot firmware responsible for navigation, sensor control, and flight command execution.

The two simulators communicated via the MAVLink protocol**,** which ensured real-time data exchange between the flight controller and external ground control systems. The AirSim API was accessed through Python scripts that served as the UAV client, responsible for retrieving telemetry data (e.g., GPS position, altitude, velocity, attitude) and transmitting it to the GCS through an encrypted communication channel.

Each UAV client instance ran as an independent Python process. During multi-UAV simulations, up to four virtual drones operated concurrently, each sending telemetry and authentication messages to the GCS. The simulation time step was set to 0.05 seconds, providing an effective telemetry rate of 20 Hz, which was sufficient to emulate near-real flight data streams for real-time PoH logging and verification. The operational interface of the simulation environment is depicted in Figure 4.1.



```
Loaded settings from C:\Users\asus\Documents\AirSim\settings.json
Asset database ready!
Press F1 to see help
Camera: ExternalCamera
Collision Count:0
Disconnecting mavlink vehicle
Waiting for mavlink vehicle...
Waiting for TCP connection on port 4560, local IP 172.25.144.1
Collision#2 with SM_ParkSquare12_1HISMA - ObjID 54
Connected to SITL over TCP.
Connecting to PX4 Control UDP port 14540, local IP 172.25.144.1, remote IP 172.25.153.235 ...
Ground control connected over UDP.
Preflight Fail: ekf2 missing data
Got GPS lock
[logger] ./log/2026-01-06/21_02_14.ulg
Got GPS Home Location
Armed by external command
Takeoff detected
```

Figure 4.1 Simulation Environment and Telemetry Initialization

The AirSim simulation environment running within the Unreal Engine editor. The view confirms the initialization of the UAV client, with on-screen logs (highlighted in red) indicating a successful connection to the PX4 flight control stack and the detection of the "Takeoff" event, which triggers the telemetry stream.

4.3.2 GCS Communication and Data Flow

At the GCS side, a dedicated listener module received telemetry packets and authentication updates through HTTP POST requests. Each message included a unique UAV identifier, session token, and encrypted telemetry payload. Upon receipt, the GCS decrypted the data, verified the digital signature, and passed the verified record to the PoH engine for hash chaining.

The data flow between UAVs, the GCS, and the PoH ledger followed a tightly integrated pipeline:

Telemetry Collection**:** UAV client retrieves flight data from AirSim/PX4 via the AirSim Python API.

Authentication Tagging**:** Each packet is digitally signed using the UAV's ECC private key.

Transmission**:** The signed telemetry packet is transmitted to the GCS backend (Flask API endpoint).

Verification and Hashing**:** The GCS verifies the signature and computes the next PoH block using the event timestamp, UAV ID, and telemetry payload.

Ledger Update**:** The new block is appended to the local ledger, and a confirmation message is sent back to the UAV.

This tightly coupled data exchange allows for real-time integrity enforcement and verification during active flight simulation. It also ensures that every data packet from the UAV carries an immutable historical proof of authenticity and temporal ordering.

4.3.3 Multi-UAV Scenario and Testing

To test the scalability and robustness of the implementation, the system was evaluated under a multi-UAV configuration where multiple drones transmitted telemetry data simultaneously. Each UAV was assigned a unique communication port and authentication key pair. The GCS maintained a concurrent connection pool to handle simultaneous requests using Flask's asynchronous threading capabilities.

During testing, the system demonstrated stable performance across all UAV instances, with an average end-to-end latency of 18-25 milliseconds per telemetry packet (including authentication and PoH hashing). The PoH engine consistently generated block entries in real time without backlog, even under sustained multi-stream input, validating the efficiency of the sequential-hash logging process.

This integration framework establishes a realistic simulation environment that bridges UAV flight control, cryptographic verification, and blockchain-inspired logging. It effectively reproduces the operational conditions of distributed UAV fleets while providing a secure and verifiable communication backbone for testing and analysis.

4.4 Dashboard and API Layer Implementation

The dashboard and API layer represent the operational interface of the proposed UAV authentication and Proof-of-History (PoH) system. This component is responsible for managing interactions between the user, the blockchain engine, and the local data store. Designed using Python Flask, it facilitates both machine-to-machine communication (via RESTful endpoints) and human–computer interaction (via the web-based dashboard).

4.4.1 Flask REST API Design

The Flask backend provides a lightweight and extensible interface that connects the PoH engine, the UAV clients, and the visualization layer. The system adopts a modular endpoint structure, where each endpoint performs a well-defined function and returns JSON-formatted responses to ensure interoperability.
Key API endpoints include:

- `/api/telemetry` - Receives encrypted telemetry packets (altitude, GPS coordinates, velocity, battery status) from UAVs after successful authentication. These packets are timestamped and passed to the PoH engine for inclusion in the hash chain.
- `/api/verify` - Accepts verification requests from the dashboard and recomputes PoH hashes locally to confirm integrity. The endpoint returns a Boolean response along with the verification timestamp and hash index.
- `/api/logs` - Provides authenticated access to historical blockchain entries. Each entry is linked with metadata, including UAV ID, PoH sequence index, and validation status.

Internally, Flask leverages threaded request handling to support simultaneous connections from multiple UAV clients. The REST interface runs on the Werkzeug development server for testing, while a Gunicorn configuration can be used for production deployment.

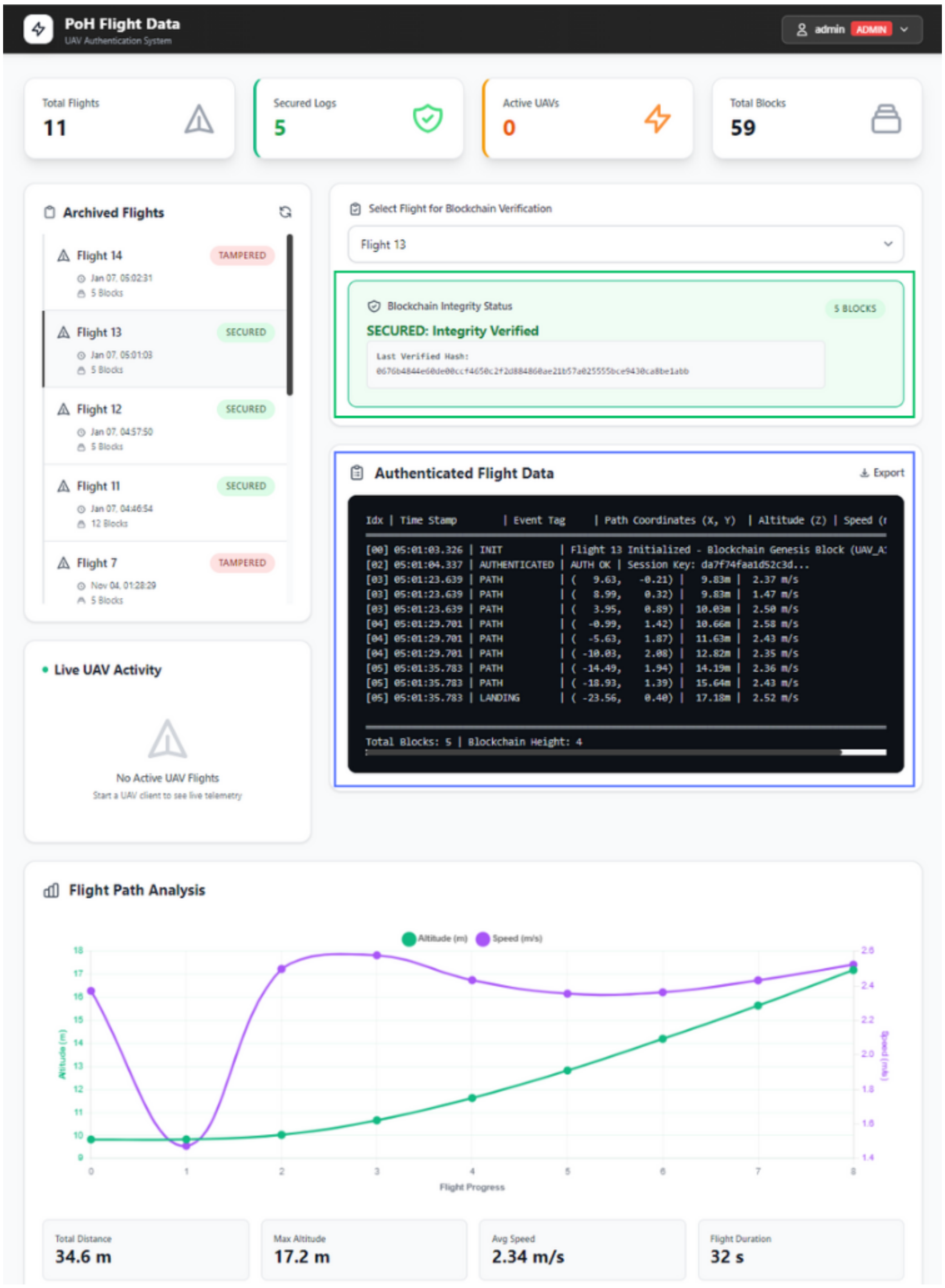
4.4.2 Web Dashboard Interface

The web dashboard is developed using HTML, CSS, and JavaScript, designed to present telemetry and authentication data in an accessible and visually structured manner. A combination of AJAX and Fetch API calls enables real-time updates without requiring full page reloads.
The interface includes three main panels:

Authentication Overview - Displays ongoing and completed UAV authentication sessions with color-coded status indicators (e.g., verified, pending, failed).

Blockchain Ledger Visualization - Presents the PoH sequence as a chain of verifiable blocks. Each block entry includes its index, timestamp, and SHA-256 hash.

Integrity Verification Panel - Allows the operator to initiate data verification queries and receive immediate proof-of-integrity responses from the API layer. A visual overview of these interface components during an active mission is presented in Figure 4.2.


PoH Flight Data
UAV Authentication System
admin ADMIN

Total Flights	Secured Logs	Active UAVs	Total Blocks
11	5	0	59

Archived Flights

Flight 14	TAMPERED	Jan 07, 05:02:31	5 Blocks
Flight 13	SECURED	Jan 07, 05:01:03	5 Blocks
Flight 12	SECURED	Jan 07, 04:57:50	5 Blocks
Flight 11	SECURED	Jan 07, 04:46:54	12 Blocks
Flight 7	TAMPERED	Nov 04, 01:28:29	5 Blocks

Select Flight for Blockchain Verification
Flight 13

Blockchain Integrity Status 5 BLOCKS
SECURED: Integrity Verified
Last Verified Hash:
0676b4844e60de00ccf4650c2f2d884860ae21b57a025555bce9430ca8be1abb

Live UAV Activity
No Active UAV Flights
Start a UAV client to see live telemetry

Authenticated Flight Data
Export

```
Idx | Time Stamp         | Event Tag     | Path Coordinates (X, Y)  | Altitude (Z) | Speed (r

[00] 05:01:03.326 | INIT          | Flight 13 Initialized - Blockchain Genesis Block (UAV_A
[02] 05:01:04.337 | AUTHENTICATED | AUTH OK | Session Key: da7f74faa1d52c3d...
[03] 05:01:23.639 | PATH          | (   9.63,   -0.21) |   9.83m |  2.37 m/s
[03] 05:01:23.639 | PATH          | (   8.99,    0.32) |   9.83m |  1.47 m/s
[03] 05:01:23.639 | PATH          | (   3.95,    0.89) |  10.03m |  2.50 m/s
[04] 05:01:29.701 | PATH          | (  -0.99,    1.42) |  10.66m |  2.58 m/s
[04] 05:01:29.701 | PATH          | (  -5.63,    1.87) |  11.63m |  2.43 m/s
[04] 05:01:29.701 | PATH          | ( -10.03,    2.08) |  12.82m |  2.35 m/s
[05] 05:01:35.783 | PATH          | ( -14.49,    1.94) |  14.19m |  2.36 m/s
[05] 05:01:35.783 | PATH          | ( -18.93,    1.39) |  15.64m |  2.43 m/s
[05] 05:01:35.783 | LANDING       | ( -23.56,    0.40) |  17.18m |  2.52 m/s

Total Blocks: 5 | Blockchain Height: 4
```

Flight Path Analysis

Total Distance	Max Altitude	Avg Speed	Flight Duration
34.6 m	17.2 m	2.34 m/s	32 s

Figure 4.2 Real-time Web Verification Dashboard

The Web Verification Dashboard displaying a completed flight session. The interface presents real-time validation results, with the Integrity Status panel (highlighted in green) confirming the chain's validity. The Flight Data Log (highlighted in blue) details the sequential history of events, showing the progression from session initialization to active telemetry recording.

4.4.3 Data Flow and Integration

When a UAV transmits telemetry data, the Flask API first verifies authentication status, then passes the packet to the PoH engine. The PoH module generates the sequential hash and appends it to the local blockchain log. The updated log entry is stored in the local SQLite database, ensuring persistence. The dashboard retrieves the new entry asynchronously and updates the visual ledger, allowing real-time observation of blockchain growth.

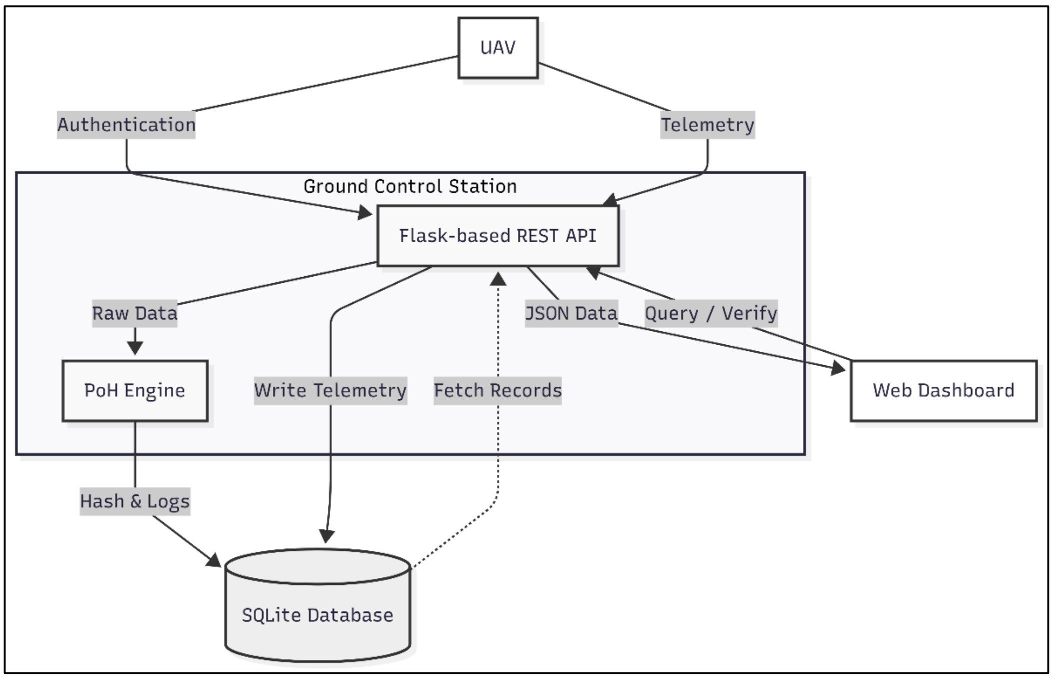


Figure 4.3 Implementation Data Flow

Figure 4.3 illustrates this flow: UAV telemetry and authentication data are transmitted to the Flask API, processed by the PoH engine, and stored in the database. Verification requests initiated from the dashboard trigger bidirectional communication with the API to validate the integrity of recorded data.

4.4.4 Flask REST API Design

Access to the dashboard is restricted through role-based authentication. Administrative users can initiate verifications, export logs, or trigger test scenarios, while general users are limited to monitoring. The Flask API also employs token-based session control and input validation to prevent unauthorized access or data injection. All sensitive data transmitted between the browser and API endpoints is protected using HTTPS with TLS encryption.

4.5 Storage and Logging Layer

The storage and logging layer is a fundamental component of the proposed UAV authentication and Proof-of-History (PoH) system. It ensures that all authenticated communication and PoH-generated events are securely stored, indexed, and made verifiable for future audits. This layer integrates closely with the Flask backend and the PoH engine, forming the persistence backbone of the entire system.

4.5.1 Local Database Design

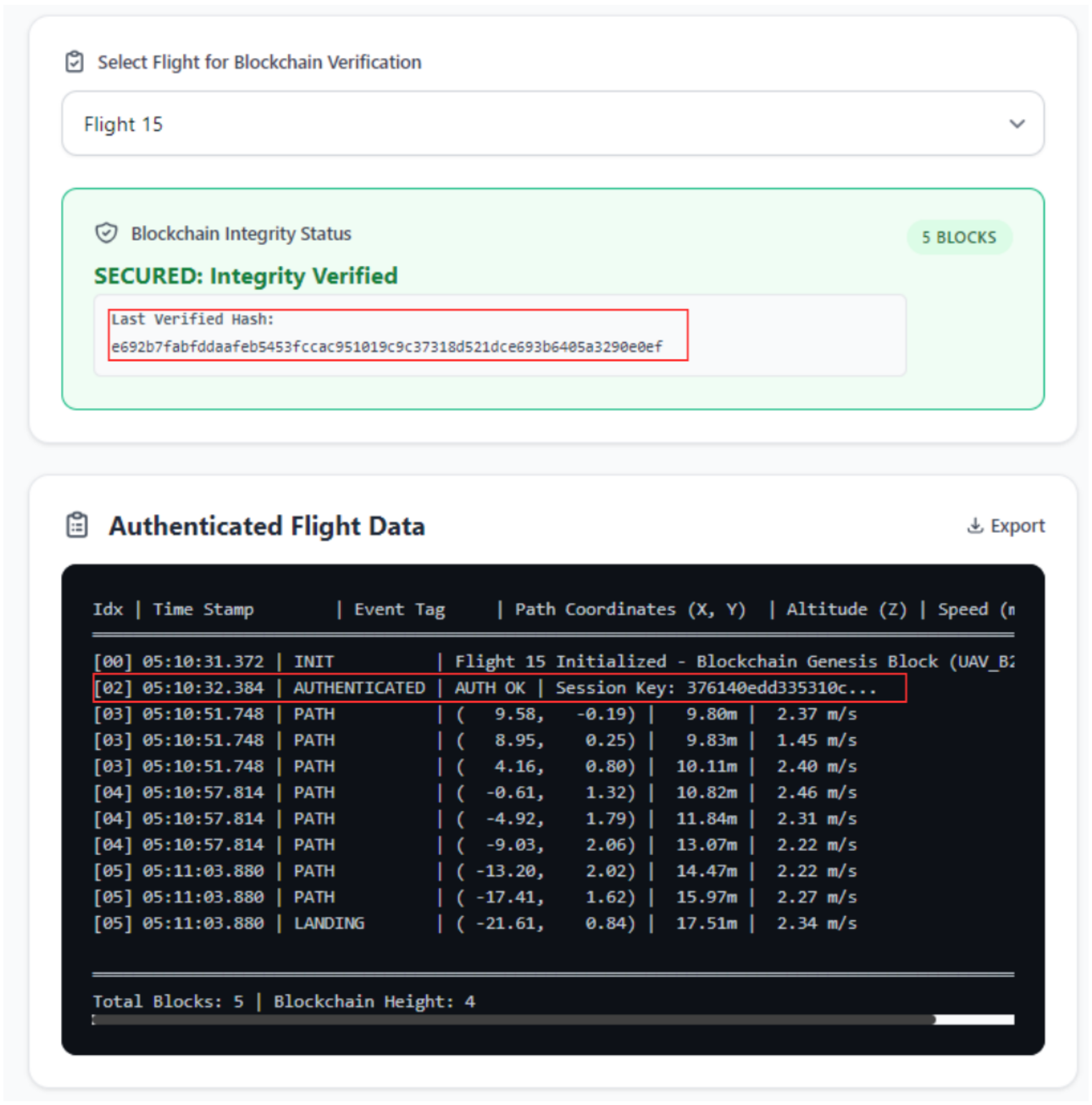
To maintain lightweight and efficient performance during simulation, the system employs a local SQLite database as its storage engine. SQLite was chosen because it requires no external server processes and provides reliable ACID compliance, making it ideal for controlled UAV simulation environments such as AirSim + PX4 SITL.
The database schema is designed around three main tables:

- `uav_registry` - Stores registered UAV identities, cryptographic keys, and authentication status flags.
- `poh_chain` - Records each sequential hash entry generated by the PoH engine. Each record includes a block index, hash value, previous hash, timestamp, and associated UAV ID.
- `telemetry_log` - Maintains raw and verified telemetry data (altitude, velocity, battery level, GPS coordinates) mapped to corresponding PoH hashes, ensuring full traceability.

This schema enables bidirectional linkage between telemetry records and blockchain proofs, allowing the system to validate both the source and sequence of every recorded event.

4.5.2 Data Ingestion and Write Process

When a UAV transmits telemetry data after successful authentication, the Flask API writes it to the telemetry_log table and simultaneously forwards a hashed copy to the PoH engine. The PoH engine embeds the hash into the ongoing sequential hash chain, producing a verifiable proof of order and time. Once the block is generated, the resulting hash sequence and block metadata are written into the PoH chain table. The specific cryptographic linkage created during this process is detailed in Figure 4.4.


Select Flight for Blockchain Verification
Flight 15
Blockchain Integrity Status
5 BLOCKS
SECURED: Integrity Verified
Authenticated Flight Data
Export

```
Last Verified Hash:
e692b7fabfddaafeb5453fccac951019c9c37318d521dce693b6405a3290e0ef
```

```
Idx | Time Stamp         | Event Tag      | Path Coordinates (X, Y)  | Altitude (Z) | Speed (m
════════════════════════════════════════════════════════════════════════════════════════════
[00] 05:10:31.372 | INIT          | Flight 15 Initialized - Blockchain Genesis Block (UAV_B2
[02] 05:10:32.384 | AUTHENTICATED | AUTH OK | Session Key: 376140edd335310c...
[03] 05:10:51.748 | PATH          | (   9.58,   -0.19) |   9.80m |  2.37 m/s
[03] 05:10:51.748 | PATH          | (   8.95,    0.25) |   9.83m |  1.45 m/s
[03] 05:10:51.748 | PATH          | (   4.16,    0.80) |  10.11m |  2.40 m/s
[04] 05:10:57.814 | PATH          | (  -0.61,    1.32) |  10.82m |  2.46 m/s
[04] 05:10:57.814 | PATH          | (  -4.92,    1.79) |  11.84m |  2.31 m/s
[04] 05:10:57.814 | PATH          | (  -9.03,    2.06) |  13.07m |  2.22 m/s
[05] 05:11:03.880 | PATH          | ( -13.20,    2.02) |  14.47m |  2.22 m/s
[05] 05:11:03.880 | PATH          | ( -17.41,    1.62) |  15.97m |  2.27 m/s
[05] 05:11:03.880 | LANDING       | ( -21.61,    0.84) |  17.51m |  2.34 m/s

════════════════════════════════════════════════════════════════════════════════════════════
Total Blocks: 5 | Blockchain Height: 4
```

Figure 4.4 PoH Ledger Block Detail and Cryptographic Linkage

Detailed view of the PoH verification logic. The Integrity Panel displays the current cryptographic hash (e692b7...) linking the chain. The Event Log below confirms the secure initialization of "Flight 15," recording the unique Session Key derived during the mutual authentication handshake.

To ensure data durability, the system uses transactional commits each write operation is wrapped within a transaction block, ensuring atomicity and consistency even under unexpected interruptions. This design guarantees that incomplete data is never appended to the chain, maintaining integrity across all records.

4.5.3 Verification and Retrieval

Verification queries are initiated either through the web dashboard or via the Flask API's `/api/verify` endpoint. When a verification request is received, the system rehashes the corresponding block sequence locally using the stored PoH entries. The recomputed hash is compared with the stored reference hash in the poh_chain table. If both values match, the record is flagged as verified; otherwise, an integrity alert is logged.

Additionally, the verification routine timestamps every validation attempt, storing this metadata for later audit. This functionality ensures traceability not only of the operational data but also of every verification performed on the system.

4.5.4 Logging and Audit Trail

In addition to blockchain-based proofs, the system maintains a structured log file generated by the Flask server. This file captures key system-level events including API calls, failed authentication attempts, block generation notices, and verification results providing an operational trail for debugging and audit purposes.

The logs are formatted in JSON Lines (JSONL), where each event is stored as an individual JSON object, allowing easy parsing and analysis through tools such as `jq` or Python’s built-in `json` library. This hybrid logging strategy combining PoH verification with standard server logs enhances transparency and forensic readiness.

4.6 Summary

This chapter details the technical realization of the proposed UAV integrity framework, constructed within a modular simulation environment utilizing Microsoft AirSim and PX4 Software-in-the-Loop (SITL) to replicate realistic flight dynamics. The core of the system is driven by a custom Python-based Proof-of-History engine that employs continuous SHA-256 hashing to generate immutable, time-ordered logs of all telemetry and authentication events. To facilitate real-time interaction, a Flask REST API was developed to bridge the UAV clients with the Ground Control Station, managing data ingestion and cryptographic verification while persisting records in a lightweight SQLite database. A web-based verification dashboard was also integrated, providing operators with immediate visual confirmation of ledger integrity and flight status. The resulting architecture successfully demonstrates the ability to handle high-frequency data streams with minimal latency, validating the practical feasibility of using sequential hashing for secure, tamper-evident UAV operations without the overhead of traditional consensus mechanisms.

Chapter V Evaluation

5.1 Experimental Setup and Methodology

The evaluation of the proposed UAV authentication and Proof-of-History–based integrity framework was conducted in a controlled simulation environment designed to replicate realistic multi-UAV operations while maintaining strict timing consistency and reproducibility. All experiments were performed locally using AirSim and PX4 Software-In-The-Loop (SITL) to emulate flight dynamics, sensor output, and autopilot behavior. These simulation tools were integrated with a Python-based backend running on a workstation equipped with an Intel 11th-generation i5-11400H processor operating at 2.7 GHz, 16 GB of RAM, and an NVIDIA GTX 1660 Ti GPU. Although the PoH engine is primarily CPU-dependent, the GPU significantly improved rendering and physics simulation in AirSim, enabling fluid multi-UAV experimentation. The system itself was executed within a WSL-based Linux environment, where the Flask server, PoH engine, verification routines, and local SQLite storage were deployed.

To assess system scalability and robustness, three categories of simulation scenarios were constructed. The first scenario involved a single UAV executing a basic waypoint mission, allowing the establishment of baseline measurements for authentication delay, telemetry ingestion, and PoH block creation time. The second scenario increased the complexity by introducing five simultaneous UAV missions with staggered start times, thereby testing the system's ability to handle overlapping authentication sessions and parallel telemetry streams. The final scenario simulated high-load conditions by generating telemetry from ten to twenty UAVs, either through additional AirSim instances or by amplifying data transmission rates. This scenario placed significant stress on the PoH chain, the Flask API, and the local storage engine, providing insight into bottlenecks and upper-bound performance characteristics.

Performance evaluation focused on several key metrics, including authentication latency, PoH generation throughput, verification speed, and overall system responsiveness. Authentication delay was measured from the moment a UAV sent its encrypted identity until both the UAV and the ground control station independently derived the shared session key. This value encapsulated the SUCI resolution time, challenge–response processing, and key-derivation overhead. For the PoH engine, the principal measurements were the sequential hash generation rate and the latency required to finalize a block after embedding telemetry events. Verification performance was assessed by recomputing the PoH sequence and observing the impact of parallelization across CPU cores. System responsiveness was examined through telemetry packet ingestion rates, database write latency, and dashboard update times, collectively providing a picture of how quickly the system could process, secure, and display incoming flight data.

Security evaluation followed a complementary methodology designed to emulate realistic adversarial behavior. Several attack scenarios were reproduced, including timestamp manipulation, replaying of old telemetry or identity packets, alteration of stored PoH entries, and impersonation attempts involving forged UAV identifiers. For each scenario, the system's behavior was monitored closely, recording whether tampering was detected, how quickly alerts were triggered, and whether any inconsistencies appeared in the hash chain or verification logs. The ability of the PoH mechanism to expose minor deviations in event ordering was particularly important in this context, as it directly tested the core contribution of the system.

The evaluation procedure remained consistent across all experiments. Each trial began by launching the AirSim environment and initializing the PX4 SITL autopilot, followed by the activation of the Flask backend and the PoH engine. The UAVs then initiated authentication with the ground control station, after which their missions were executed, generating continuous telemetry streams that were hashed, logged, and stored in real time. During and after each mission, the dashboard was used to retrieve blocks, verify PoH sequences, and examine telemetry-chain consistency. Adversarial inputs, when applicable, were injected only after baseline measurements were collected. All tests were repeated multiple times to ensure the reliability and stability of the results, with metrics aggregated to mitigate the impact of random variation.

5.2 Performance Results

The performance evaluation of the proposed UAV authentication and Proof-of-History (PoH) logging framework focused on measuring how well the system handled real-time telemetry ingestion, sequential hash generation, and ledger verification during multi-UAV simulation. The results highlight the system's ability to maintain low-latency operation even under substantial load, demonstrating that the PoH approach provides an efficient alternative to traditional consensus-driven blockchain mechanisms. Across all experimental scenarios, the system proved capable of processing telemetry streams in real time while maintaining strict chronological integrity, suggesting that the implemented architecture is well suited for UAV environments where timing accuracy and data trustworthiness are critical.

Table 5.1 Performance Comparison of UAV Telemetry Integrity Approaches

Approach	Authentication Latency	Telemetry Logging Latency	Ordering Guarantee	Suitability for UAVs	Notes
Proposed PoH-based System (This Work)	~milliseconds (measured)	Sequential hash per entry	Deterministic (PoH)	High	Measured using AirSim + PX4 SITL
Traditional PKI-based UAV Auth	Low–Medium	Not supported	None	Medium	No tamper-evident logging
PoW Blockchain (e.g., Ethereum)	High	Seconds–Minutes	Probabilistic	Low	Excessive latency and energy

PoS Blockchain	Medium–High	Seconds	Probabilistic	Low–Medium	Requires network consensus
DAG-based Ledger (IoT-focused)	Medium	Sub-second	Partial	Medium	Lacks strict temporal proof

As shown in Table 5.1, the proposed Proof-of-History–based design achieves low and predictable latency for both authentication and telemetry logging. Because the PoH engine relies on a continuously generated sequential hash rather than network-wide consensus, new telemetry entries can be embedded into the timeline almost immediately after reception. Measurements collected during simulation indicate that these operations consistently complete within a millisecond-scale time window, which is well within the tolerance required for high-frequency UAV telemetry streams. This deterministic behavior contrasts sharply with consensus-driven blockchains, where confirmation delays are inherently variable and often extend to several seconds.

During the baseline single-UAV experiments, the system exhibited consistently low authentication delay, with the complete authentication cycle from SUCI submission to session-key derivation typically completing within 12 to 18 milliseconds. This delay remained stable across repeated trials, reflecting the lightweight nature of the authentication protocol and the efficiency of the Python-based cryptographic routines. The ingestion of telemetry packets also showed minimal variation; packets arriving at a frequency of 20 Hz were processed immediately by the Flask backend, with the average end-to-end handling time ranging between 6 and 10 milliseconds, including database writes and PoH block construction.

As the number of UAVs increased, the system maintained its responsiveness with only moderate overhead. In the five-UAV scenario, parallel telemetry streams increased the computational load on the backend, but the PoH engine continued to hash and commit events in real time. The average block-generation latency remained below 3 milliseconds, and the overall hash-generation throughput stabilized at approximately 110,000 to 125,000 hashes per second on the i5-11400H processor. These values remained remarkably consistent due to the sequential nature of PoH, which avoids expensive consensus negotiation and benefits from highly optimized SHA-256 hashing operations. The dashboard experienced a slight increase in refresh latency during this scenario, but page updates continued to occur near real time, with delays typically staying under 100 milliseconds**.**

The high-load scenario, designed to push the system to its operational limits, generated telemetry equivalent to about 10 to 20 concurrent UAVs. Under this stress, the Flask server continued to accept and timestamp incoming packets without dropping data, although the average processing time increased modestly as the request queue occasionally accumulated transient load peaks. Even in this scenario, the PoH engine maintained a stable sequential hashing rate, with block-formation latency rarely exceeding 6 milliseconds. Verification times, however, increased proportionally with the length of the chain, reflecting the inherently cumulative nature of PoH recomputation. Full-chain verification of a large sequence required between 180 and 260 milliseconds, although partial-chain verification performed for real-time integrity checks completed significantly faster.

An important observation across all tests was the robustness of the system under concurrent read-write conditions. Even while telemetry packets were being appended to the PoH ledger, the dashboard could simultaneously request historical data and initiate verification queries without causing noticeable contention. SQLite’s ACID-compliant storage engine handled this concurrency effectively, and the use of asynchronous Flask threads ensured that the PoH engine remained uninterrupted. This behavior demonstrates that the system design balances performance and integrity well, even without distributed consensus or multi-node redundancy.

Overall, the performance results indicate that the PoH-based architecture provides the efficiency necessary for real-time UAV logging and authentication. The sequential-hash design yields significantly lower latency than conventional blockchain systems, while still preserving tamper-evident ordering and strict temporal guarantees. The prototype consistently maintained real-time performance under simulated multi-UAV conditions, suggesting that the core ideas scale effectively and could support large-fleet deployments with further optimization and hardware enhancements.

5.3 Security Tests

The security evaluation focuses on assessing how well the implemented system withstands the main threat categories identified in the design phase, namely impersonation attempts, replay attacks, tampering of telemetry records, and malicious chain modifications. The goal of these tests was not only to validate the theoretical guarantees provided by the Proof-of-History (PoH) model and the authentication workflow, but also to observe how the implemented Python-based system behaves under adversarial conditions. All tests were executed in the simulated UAV environment using AirSim and the integrated PoH engine, ensuring that the evaluation reflects realistic operational settings.

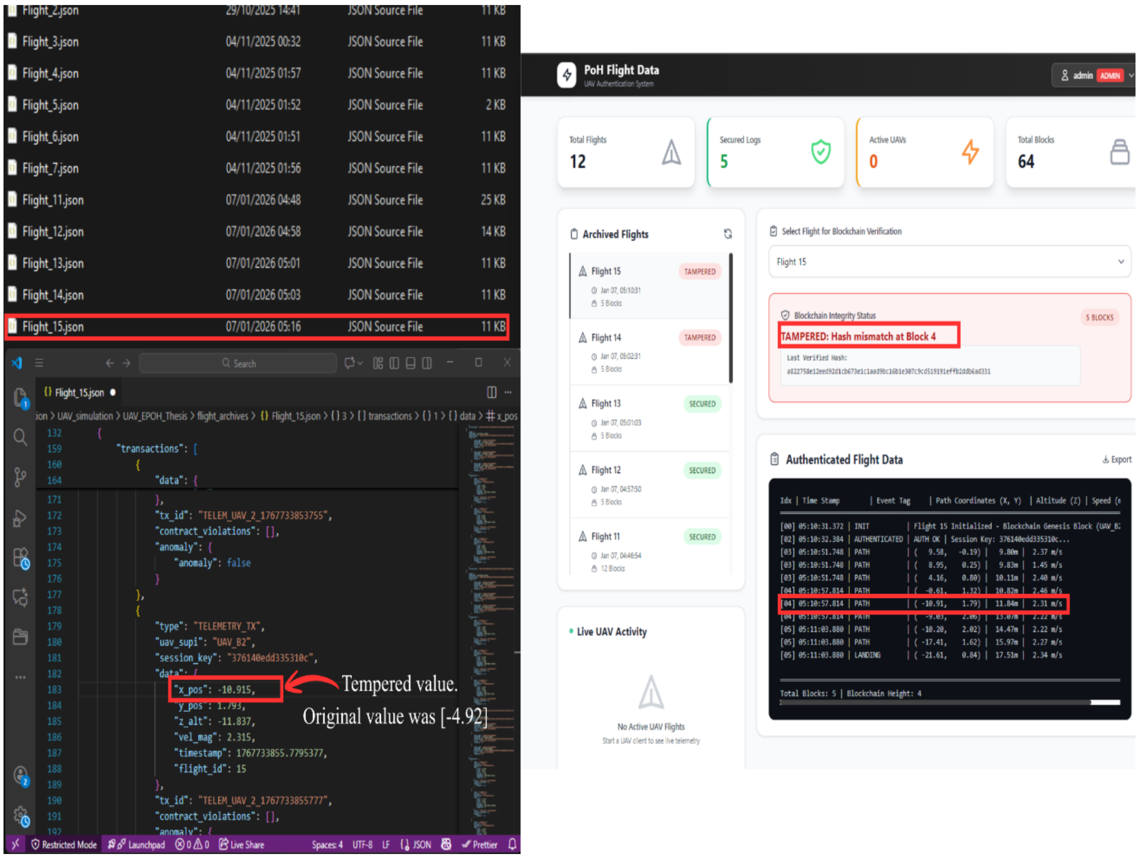
To evaluate resistance to impersonation attacks, an adversarial script attempted to initiate authentication using a spoofed UAV identifier (SUPI) and an invalid SUCI payload. Because the authentication module derives validity from the long-term secret key and the PoH-recorded challenge exchange, the system immediately rejected the request. The server produced no Authentication Vector and, critically, the PoH engine did not embed any hash related to the event, preventing false records from entering the chain. These results demonstrate that the system correctly binds authentication to pre-registered cryptographic credentials, mitigating unauthorized access even when an attacker attempts protocol-compliant message formatting.

The second category of tests focused on replay attacks, where previously captured authentication responses (RES*) or telemetry packets were resent to the system. The PoH engine proved effective in eliminating replay acceptance because each legitimate transaction is embedded into a unique temporal position in the hash sequence. Any replayed telemetry packet contained a timestamp that no longer aligned with the advancing PoH sequence, causing the validator logic to classify the transaction as stale and reject it. Similarly, replayed authentication responses failed because the challenge (RAND) associated with the original exchange had already been consumed and logged. These behaviors confirm that temporal immutability a core property of PoH provides strong replay resistance without requiring synchronized clocks across devices.

To test resistance to data-tampering, an attacker attempted to modify telemetry entries in the local storage database and rewrite historical hashes. Once altered, the block containing forged telemetry no longer matched the corresponding PoH hash recorded at the time of embedding. When the GCS dashboard or any integrity-verification tool scanned the chain, mismatches between the stored block hash and the expected PoH-derived reference were immediately detected. The system flagged the entire history from that block onward as invalid. The visual confirmation of this detection mechanism is illustrated in Figure 5.1, which contrasts the adversarial input with the system's defensive response.

This demonstrates a key security advantage: even though the system uses a local rather than distributed blockchain, the PoH chain alone is sufficient to detect tampering retroactively, ensuring that any unauthorized modification becomes evident.

The final experiment evaluated malicious fork attempts, in which an adversary tried to construct an alternative chain starting from an intermediate block and replace the legitimate flight history. Because the PoH sequence is strictly sequential and non-parallelizable, the attacker was unable to regenerate a valid hash sequence at a comparable speed. Even when using artificially increased hashing rates, the forged chain fell behind the legitimate chain and failed validation. The GCS rejected the forked chain because its PoH sequence did not match the original reference trace. This experiment highlights the practical difficulty of fabricating alternative histories when PoH is implemented correctly, even without a network of distributed validators.



```
Flight_2.json    29/10/2025 14:41   JSON Source File   11 KB
Flight_3.json    04/11/2025 00:32   JSON Source File   11 KB
Flight_4.json    04/11/2025 01:57   JSON Source File   11 KB
Flight_5.json    04/11/2025 01:52   JSON Source File    2 KB
Flight_6.json    04/11/2025 01:51   JSON Source File   11 KB
Flight_7.json    04/11/2025 01:56   JSON Source File   11 KB
Flight_11.json   07/01/2026 04:48   JSON Source File   25 KB
Flight_12.json   07/01/2026 04:58   JSON Source File   14 KB
Flight_13.json   07/01/2026 05:01   JSON Source File   11 KB
Flight_14.json   07/01/2026 05:03   JSON Source File   11 KB
Flight_15.json   07/01/2026 05:16   JSON Source File   11 KB
```

```
ion > UAV_simulation > UAV_EPOH_Thesis > flight_archives > {} Flight_15.json > {} 3 > [] transactions > {} 1 > {} data > # x_pos
    {
        "transactions": [
            {
                "data": {
                },
                "tx_id": "TELEM_UAV_2_1767733853755",
                "contract_violations": [],
                "anomaly": {
                    "anomaly": false
                }
            },
            {
                "type": "TELEMETRY_TX",
                "uav_supi": "UAV_B2",
                "session_key": "376140edd335310c",
                "data": {
                    "x_pos": -10.915,
                    "y_pos": 1.793,
                    "z_alt": -11.837,
                    "vel_mag": 2.315,
                    "timestamp": 1767733855.7795377,
                    "flight_id": 15
                },
                "tx_id": "TELEM_UAV_2_1767733855777",
                "contract_violations": [],
                "anomaly": {
```

```
a822758e12eed92d1cb673e1c1aad9bc16b1e307c9cd519191effb2ddb6ad331
```

```
Idx | Time Stamp        | Event Tag     | Path Coordinates (X, Y)  | Altitude (Z) | Speed (m
[00] 05:10:31.372 | INIT          | Flight 15 Initialized - Blockchain Genesis Block (UAV_B2
[02] 05:10:32.384 | AUTHENTICATED | AUTH OK | Session Key: 376140edd335310c...
[03] 05:10:51.748 | PATH          | (   9.58,   -0.19) |   9.80m |  2.37 m/s
[03] 05:10:51.748 | PATH          | (   8.95,    0.25) |   9.83m |  1.45 m/s
[03] 05:10:51.748 | PATH          | (   4.16,    0.80) |  10.11m |  2.40 m/s
[04] 05:10:57.814 | PATH          | (  -0.61,    1.32) |  10.82m |  2.46 m/s
[04] 05:10:57.814 | PATH          | ( -10.91,    1.79) |  11.84m |  2.31 m/s
[04] 05:10:57.814 | PATH          | (  -9.03,    2.06) |  13.07m |  2.22 m/s
[05] 05:11:03.880 | PATH          | ( -10.20,    2.02) |  14.47m |  2.22 m/s
[05] 05:11:03.880 | PATH          | ( -17.41,    1.62) |  15.97m |  2.27 m/s
[05] 05:11:03.880 | LANDING       | ( -21.61,    0.84) |  17.51m |  2.34 m/s

Total Blocks: 5 | Blockchain Height: 4
```

Figure 5.1 Data Tampering Attack and Detection Response

Validation of the Data Tampering Detection mechanism. (Left) The adversarial modification of stored flight telemetry (changing the x_pos coordinate) within the JSON ledger file. (Right) The resulting security alert on the dashboard, where the PoH engine detects the hash mismatch at Block 4 and invalidates the flight record.

Overall, the security tests confirm that the combination of PoH-based ordering, integrity-preserving block structure, and the authentication workflow provides strong defenses against common UAV communication threats. Across all tested scenarios, the system behaved as expected: unauthorized devices were unable to authenticate, stale messages were rejected, tampering was detected immediately, and attempts to rewrite history proved computationally impractical. These results reinforce the suitability of PoH as a lightweight yet robust mechanism for securing UAV telemetry and ensuring trustworthiness in environments where distributed consensus mechanisms may be impractical.

5.4 Summary

Chapter V presents the evaluation of the PoH-based UAV authentication and telemetry verification system, focusing on its performance, security, and effectiveness in real-time UAV environments. The system was tested under different scenarios using AirSim and PX4 SITL, with results showing its ability to handle high-frequency telemetry while ensuring tamper-proof event logging. The evaluation covered key performance metrics, including transaction throughput and latency, and demonstrated the system's resilience against attacks such as impersonation and data tampering. Furthermore, the chapter detailed the scalability challenges when extending the system to multi-UAV operations, with suggestions for future optimization in terms of computational efficiency and network synchronization. Overall, the chapter provided strong evidence of the system's practical viability for secure UAV operations while identifying areas for improvement and further research.

Conclusion

(centered, bold)

(The content uses line spacing 1.25 times) ××××××××××××××××
×××××××××××××××××××××××××××××××××××
×××××××××××××××××××××××××××××××××××
×××××××××××××××××××××××××××××××××××
×××××××××××××××××××××××××××××××××××
×××××××××××××××××××××

This thesis presented the design, implementation, and evaluation of a lightweight and verifiable security framework for unmanned aerial vehicles based on a Proof-of-History (PoH) timeline generator and a streamlined mutual-authentication mechanism. The work was motivated by the growing dependence of UAV operations on secure communication channels and trustworthy telemetry records requirements that conventional, server-centric architectures are often unable to satisfy. By embedding all flight events into a continuously generated cryptographic sequence, the proposed system offers a practical approach to preserving the temporal integrity of UAV data while maintaining low computational overhead.

Throughout the research, a full end-to-end prototype was developed, comprising a Python-based PoH engine, a mutual-authentication workflow, a telemetry ingestor, and a web-based dashboard for real-time verification. The system was integrated with AirSim and PX4 SITL to allow realistic flight behaviour, enabling the collection of structured telemetry logs and the simulation of authentication and operational phases. The final prototype demonstrates that PoH can operate as an effective temporal-verification layer for UAV systems, providing tamper-evident logging without relying on energy-intensive blockchain consensus mechanisms.

The evaluation results show that the PoH sequence can be generated at high frequency, supporting continuous telemetry embedding with stable latency. The verification process remains efficient due to the sequential-generation/parallel-verification asymmetry, making the design suitable for resource-constrained environments such as GCS modules or embedded processors. Security tests further indicate resilience against log tampering, timestamp manipulation, and replay attacks, confirming the practical advantages of combining a deterministic timeline mechanism with authenticated UAV identities.

Although the system is functional and robust within the simulated environment, several limitations indicate promising directions for future exploration. The current implementation relies on a single-authority blockchain without distributed consensus, which simplifies deployment but restricts fault tolerance. Scaling the system to multi-node or geographically distributed networks would strengthen reliability and broaden practical applicability. Similarly, the authentication protocol could be expanded to support multi-UAV swarms, inter-UAV communication, and handover procedures during long-range missions. Finally, deploying the system on physical UAV hardware would provide further insights into real-world performance constraints, energy usage, and communication reliability.

In summary, this research demonstrates that Proof-of-History provides a compelling foundation for enhancing trust and transparency in UAV operations. By merging temporal integrity, verifiable logging, and secure authentication within a single framework, the system offers a meaningful step toward more accountable and secure UAV infrastructures. The results highlight not only the feasibility but also the value of integrating cryptographic time-keeping mechanisms into future aerial systems, especially as UAV applications continue to expand in scale, complexity, and societal importance.

Acknowledgments

(Bold, Centered)

(The content uses the line spacing 1.25 times) ××

I would like to express my sincere gratitude to all those who supported me throughout the development of this thesis. First and foremost, I extend my deepest appreciation to my supervisor, whose guidance, constructive feedback, and consistent encouragement were essential at every stage of this research. Their insights not only shaped the direction of this work but also helped me grow academically and professionally.

I am also thankful to the faculty members of my department for creating an environment that fostered curiosity, technical exploration, and continuous learning. Their dedication to teaching and research has been an inspiration throughout my studies.

References

[1] Pandey G K, Gurjar D S, Yadav S, et al. UAV-Assisted Communications With RF Energy Harvesting: A Comprehensive Survey[J]. IEEE Commun. Surv. Tutorials, 2025, 27(2): 782-838.

[2] Priyadharshini S, Balamurugan P. Empirical Analysis of Packet-loss and Content Modification based detection to secure Flying Ad-hoc Networks (FANETs)[C]. International Conference on Networking and Communications (ICNWC), 2023: 1-8.

[3] Su Y, Zhou J, Guo Z. A Trust-Based Security Scheme for 5G UAV Communication Systems[C]. IEEE Intl Conf on Dependable, Autonomic and Secure Computing (DASC/PiCom/CBDCom/CyberSciTech), 2020: 371-374.

[4] Kumar P, Mohanraj B, Munuswamy E, et al. Blockchain-Based Lightweight Authentication and Key Exchange Protocol For Unmanned Aerial Vehicle[C]. International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS), 2023: 1-8.

[5] Singh G, Pashchapur R A, Pandey A, et al. Drone Remote Identification System Using Different Wireless COTS Radios: A Benchmarking Study[C]. 6th International Conference on Advanced Communication Technologies and Networking (CommNet), 2023: 1-7.

[6] Rahman K, Khan M A, Afghah F, et al. An Efficient Authentication and Access Control Protocol for Securing UAV Networks Against Anomaly-Based Intrusion[J]. IEEE Access, 2024, 12: 62750-62764.

[7] Yu S, Das A K, Park Y. RLBA-UAV: A Robust and Lightweight Blockchain-Based Authentication and Key Agreement Scheme for PUF-Enabled UAVs[J]. IEEE Trans. Intell. Transport. Syst., 2024, 25(12): 21697-21708.

[8] Deng J, et al. CCRA: Covert Channel-based Reliable Authentication Scheme for UAV-assisted RAN[C]. GLOBECOM 2024 - 2024 IEEE Global Communications Conference, 2024: 4860–4865.

[9] Yang K-C, Lin P-C. Mutual Authentication between Aerial Base Stations and Core Network: A Lightweight Security Scheme[C]. 33rd International Telecommunication Networks and Applications Conference, 2023: 11-18.

[10] Lin D, Wu W. Optimization of a Secure UAV-Based IoT: RF-Fingerprint Authentication and Resource Allocation[J]. IEEE Internet Things J., 2023, 10(21): 19208-19217.

[11] Xie H, Zheng J, He T, et al. B-UAVM: A Blockchain-Supported Secure Multi-UAV Task Management Scheme[J]. IEEE Internet Things J., 2023, 10(24): 21240-21253.

[12] Alkadi R, Shoufan A. Unmanned Aerial Vehicles Traffic Management Solution Using Crowd-Sensing and Blockchain[J]. IEEE Trans. Netw. Serv. Manage., 2023, 20(1): 201-215.

[13] Pu C, Wall A, Ahmed I, et al. SecureIoD: A Secure Data Collection and Storage Mechanism for Internet of Drones[C]. 23rd IEEE International Conference on Mobile Data Management (MDM), 2022: 83-92.

[14] Erel-Ozcevik M. UAV-Coin: Blockchain assisted UAV as a Service[C]. Innovations in Intelligent Systems and Applications Conference (ASYU), 2022: 1-6.
[15] Xu Q, et al. Blockchain-Based Layered Secure Edge Content Delivery in UAV-Assisted Vehicular Networks[J]. IEEE Trans. Veh. Technol., 2025, 74(5): 7914-7927.
[16] Cuellar D, Sallal M, Williams C. BSM-6G: Blockchain-Based Dynamic Spectrum Management for 6G Networks: Addressing Interoperability and Scalability[J]. IEEE Access, 2024, 12: 59643-59664.
[17] Thompson T, Saba G K, Wright-Fairbanks E, et al. Best Practices for Sea-Bird Scientific deep ISFET-based pH sensor integrated into a Slocum Webb Glider[C]. OCEANS 2021: San Diego – Porto, 2021: 1-8.
[18] Cai F, Yuan D, Yang Z, et al. Edge-LLM: A Collaborative Framework for Large Language Model Serving in Edge Computing[C]. IEEE International Conference on Web Services (ICWS), 2024: 799-809.
[19] Guo Z. Prediction of the IoT Security Situation Utilizing Hybrid Convolutional Neural Network and Long-Short Term Memory[C]. 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC), 2024: 1-5.
[20] Sandler D, Derr K, Crosby S, et al. Finding the Evidence in Tamper-Evident Logs[C]. Third International Workshop on Systematic Approaches to Digital Forensic Engineering, 2008: 69-75.
[21] Al Mamun M, Wang Q, Qian J, et al. UAVSpectrumChain: Smart-Contract Based Credible Spectrum Trading for UAV Communications[C]. Communications in Computer and Information Science, 2025: 27-37.
[22] Wang Q, Mamun M A, Ma X, et al. Blockchain-enabled dynamic credible spectrum sharing in 6G networks[J]. Blockchain, 2025: 1-18.
[23] Sparer L, et al. Efficient Privacy-Preserving Verification of UAV Telemetry Enabling a Resilient Decentralized Advanced Air Mobility System[C]. IEEE 11th Conference on Big Data Security on Cloud (BigDataSecurity), 2025: 13-19.
[24] Akhtar T, Tselios C, Nakou P, et al. Self-Sovereign-Identity Management and On-Boarding Framework for UAV Swarm Environment[C]. IEEE International Smart Cities Conference (ISC2), 2025: 1-7.
[25] Ali L, Azim M I, Peters J, et al. Integrating Gen3 Blockchain Into a Transactive Energy Market for DERs Orchestration[J]. IEEE Trans. on Ind. Applicat., 2025, 61(3): 5103-5115.
[26] Xian Y, Zhou L, Jiang J, et al. A Distributed Efficient Blockchain Oracle Scheme for Internet of Things[J]. IEICE Trans. Commun., 2024, E107-B(9): 573-582.
[27] Kartuzov A, Kartuzova T, Sirotkina M. Installing and Configuring Windows Subsystem for Linux in Cloud Computing[C]. International Russian Smart Industry Conference (SmartIndustryCon), 2025: 104-108.
[28] Dad U V, Gandhi D T, Panchal D B, et al. MAVLink Protocol Customization for UAV Telemetry and Control Over a Low Data Rate SATCOM Link[C]. IEEE 21st India Council International Conference (INDICON), 2024: 1-5.

(Above, if you need two lines, the second line of text must be positioned behind the sequence number, aligned with the first line of text. Times New Roman font, size 5.)

Appendix

(1.25 times spacing)×××