Proof of Location Consensus Network: A Peer-to-Peer System for Secure Aviation Surveillance

Richard Walsh, MSc., A.A.E, a3i global aeronautics advisors, LLC

Abstract

Automatic Dependent Surveillance-Broadcast (ADS-B) is a foundational technology for modern air traffic management, enabling real-time position sharing among aircraft and ground stations. However, its open, unauthenticated broadcasts are vulnerable to spoofing, jamming, and interception, posing risks to safety and scalability in Advanced Air Mobility (AAM) and Unmanned Aircraft Systems (UAS). I propose a peer-to-peer network employing a proof-of-location consensus mechanism to mitigate these vulnerabilities. Aircraft act as nodes, broadcasting tokenized ADS-B data that is validated through a distributed ledger. Consensus is achieved not by computational work, but by verifiable location proofs derived from multi-source signals (e.g., GNSS, INS, and multilateration). This creates an immutable chain of location records, resistant to tampering, while incentivizing participation via operational credits. The system operates without reliance on central authorities, enhancing reliability in dense, autonomous airspace.

Introduction

Commerce in the skies relies on trust: trust that reported aircraft positions are accurate, that signals are genuine, and that malicious actors cannot disrupt operations. Traditional ADS-B, while efficient, broadcasts unencrypted data on open frequencies (978 MHz and 1090 MHz), allowing easy spoofing—inserting false aircraft—or jamming to overwhelm receivers. Interception further enables tracking without consent, undermining privacy and security.

Current mitigations, such as machine learning for anomaly detection or encryption, introduce latency or complexity incompatible with real-time aviation needs. What is needed is a system where aircraft collectively validate each other's locations in a decentralized manner, ensuring integrity without a trusted intermediary like ground-based ATC.

I introduce the Proof of Location Consensus Network (PoLCN), a blockchain-inspired protocol where ADS-B messages are tokenized and appended to a distributed ledger. Validation occurs via proof-of-location, leveraging physical signals and consensus among

nearby nodes. This mirrors the Bitcoin network's proof-of-work but replaces energy-intensive mining with location-based verification, suitable for low-power avionics.

The network runs on equipped aircraft, with ground stations as optional validators. It scales with airspace density, providing immutable audit trails for post-incident analysis and enabling autonomous operations in UAM ecosystems.

ADS-B Tokens

I define an ADS-B "token" as a digital representation of an aircraft's broadcast message. Each token includes:

- ICAO aircraft address (24 bits)
- Type Code (TC) and data payload (e.g., position, velocity, altitude)
- Timestamp from GNSS or INS
- Hash of the previous token in the chain

Tokens are owned by the broadcasting aircraft but validated collectively. Ownership transfer isn't literal; instead, tokens are "spent" by appending new data, creating a chain of location history.

A typical token structure:

```
{
 "icao": "A12345",
 "tc": 9,  // Airborne position with barometric altitude
 "data": {
  "lat": 37.7749,
  "lon": -122.4194,
  "alt": 10000,
  "vel": 450
 },
 "timestamp": 1722432000,
 "prev_hash": "000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
 "signature": "ecdsa_signature_here"
}
```

Tokens are signed using ECDSA (Elliptic Curve Digital Signature Algorithm) with private keys stored in secure avionics modules, preventing forgery.

Timestamp Server

To prevent double-broadcasting or replay attacks, I employ a distributed timestamp server. Each token is hashed and broadcast to the network. Nodes collect tokens into blocks,

timestamp them via a consensus clock synchronized to UTC (from GNSS), and hash the block.

This creates a chain where each block references the previous one's hash, forming an immutable timeline of airspace events.

Proof-of-Location

The core innovation is proof-of-location (PoL), a consensus mechanism replacing proof-of-work. To add a block, a node must prove its claimed location matches observed signals from peers.

Steps:

1. A node broadcasts a candidate block of tokens.
2. Nearby nodes (within radio range) verify the broadcaster's location using:
    a. Time Difference of Arrival (TDoA): Measure signal propagation delays.
    b. Signal strength correlation: Compare received power to expected (based on TX power, typically 20-250W).
    c. Cross-validation with own sensors (e.g., INS estimates vs. reported GNSS).
3. Consensus: At least 51% of validating nodes must agree on the location's validity. Validators stake operational credits (see Incentives) on their votes; incorrect validations forfeit stakes.

PoL difficulty adjusts based on airspace density: higher in urban areas for robustness against jamming.

Pseudocode (python) for PoL validation:

```
import hashlib
import time
from ecdsa import SigningKey, VerifyingKey

def calculate_tdoa(signal_time, expected_time):
  return abs(signal_time - expected_time) < TDOA_THRESHOLD  # e.g., 1ms

def validate_location(token, observed_signals):
  # Hash token data
  token_hash = hashlib.sha256(str(token).encode()).hexdigest()

  # Verify signature
  vk = VerifyingKey.from_string(token['public_key'])
  if not vk.verify(token['signature'], token_hash.encode()):
    return False

  # Check TDoA and signal strength
  for signal in observed_signals:
    if not calculate_tdoa(signal['rx_time'], token['timestamp'] + propagation_delay(signal['distance'])):
      return False
    if signal['rx_power'] < MIN_SIGNAL_STRENGTH:  # Anti-jamming check
      return False

  return True

def achieve_consensus(candidate_block, network_nodes):
  votes = []
  for node in network_nodes:
    if validate_location(candidate_block['tokens'][0], node['observations']):  # Simplify for first token
      votes.append(node['vote'])

  if sum(votes) / len(votes) > 0.51:
    return True
  return False
```

This ensures only genuine, location-verified blocks are added, mitigating spoofing (false positions fail TDoA) and jamming (low signal strength flags interference).

Network

The network forms ad-hoc via ADS-B frequencies, augmented by optional data links (e.g., 5G for ground integration). Steps to run:

1. Nodes discover peers via periodic beacons.
2. New tokens are broadcast and gossiped.
3. Longest valid chain (by PoL-verified blocks) is accepted.
4. Orphaned branches (e.g., from jammed regions) are discarded.

In sparse airspace, fallback to ground validators or satellite links maintains connectivity.

Sample Python code for a basic node simulator:

```
import socket
import threading
import json

class PoLCNNode:
  def __init__(self, host, port, icao):
    self.host = host
    self.port = port
    self.icao = icao
    self.chain = []  # Genesis block
    self.peers = []
    self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

  def start(self):
    threading.Thread(target=self.listen).start()

  def listen(self):
    self.sock.bind((self.host, self.port))
    while True:
      data, addr = self.sock.recvfrom(1024)
      message = json.loads(data)
      if message['type'] == 'token':
        self.process_token(message['data'])
      elif message['type'] == 'block':
        if achieve_consensus(message['block'], self.peers):  # From section 4
          self.chain.append(message['block'])

  def broadcast_token(self, token):
    for peer in self.peers:
      self.sock.sendto(json.dumps({'type': 'token', 'data': token}).encode(), peer)

  def process_token(self, token):
    # Validate and add to pending, mine block if validator
    pass  # Implement mining with PoL

# Example usage
node = PoLCNNode('127.0.0.1', 5000, 'A12345')
node.start()
```

This is a simplified REPL-friendly implementation; real deployment would use avionics-certified languages like Ada.

Incentive

Nodes are incentivized by "location credits"—tokens earned for successful validations. Credits can be redeemed for priority routing in congested airspace or reduced fees in UTM systems. Malicious nodes lose credits, discouraging attacks.

Validators stake credits on blocks; honest behavior yields rewards proportional to stake and airspace contribution.

Reclaiming Resources

Chains grow with traffic, but avionics have limited storage. Prune old blocks after a horizon (e.g., 1 hour), retaining Merkle roots for verification. Full archives on ground nodes.

Simplified Validation

Light nodes (small UAS) verify headers only, querying full nodes for Merkle proofs of specific tokens.

Privacy

Tokens use zero-knowledge proofs for position sharing: reveal relative proximity without absolute coordinates, preserving anonymity in non-critical scenarios.

Calculations

Assume an attacker controls <51% of local nodes. Probability of spoofing a block:

Let p = honest validation probability, q = 1-p.

Attacker succeeds if q > 0.5, but PoL ties validation to physics, making q low in dense areas.

For jamming: Signal must overpower legitimate broadcasts, but PoL flags anomalies, isolating the jammer.

Conclusion

I have proposed a method to secure ADS-B using a proof-of-location blockchain, enabling trustless, resilient aviation surveillance. By leveraging existing signals for consensus, PoLCN mitigates key vulnerabilities without overhauling infrastructure. This paves the way for scalable AAM, where aircraft autonomously collaborate in a tamper-proof network.

References

[1] Strohmeier et al., "On the Security of the Automatic Dependent Surveillance-Broadcast Protocol," IEEE Comm. Surveys & Tutorials, 2015.
[2] Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
[3] FAA, "Automatic Dependent Surveillance-Broadcast (ADS-B)," 2023.
[4] Walsh, "Blockchain for Aviation Operations Concept," 2023.