Advances in Knowledge-Based Systems, Data Science, and Cybersecurity Received 27-04-2025; Accepted 19-06-2025; Published 26-06-2025

ZAPS: A Zero-Knowledge Proof Protocol for Secure UAV Authentication with Flight Path Privacy

Shayesta Naziri Shayesta.Naziri@student.uts.edu.au
School of Electrical and Data Engineering, University of Technology Sydney
NSW, 2007, AUS

Xu Wang Xu.Wang@uts.edu.au
School of Electrical and Data Engineering, University of Technology Sydney
NSW, 2007, AUS

Guangsheng Yu Guangsheng.Yu@uts.edu.au
School of Electrical and Data Engineering, University of Technology Sydney
NSW, 2007, AUS

Christy Jie Liang Jie.Liang@uts.edu.au
School of Computer Science, University of Technology Sydney
NSW, 2007, AUS

Wei Ni wei.ni@ieee.org
School of Electrical and Data Engineering, University of Technology Sydney
NSW, 2007, AUS

Corresponding Author: Shayesta Naziri

Abstract

The increasing deployment of Unmanned Aerial Vehicles (UAVs) for military, commercial, and logistics applications has raised significant concerns regarding flight path privacy. Conventional UAV communication systems often expose flight path data to third parties, making them vulnerable to tracking, surveillance, and location inference attacks. Existing encryption techniques provide security but fail to ensure complete privacy, as adversaries can still infer movement patterns through metadata analysis. To address these challenges, we propose a zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge)-based privacy preserving flight path authentication and verification framework. Our approach ensures that a UAV can prove its authorisation, validate its flight path with a control centre, and comply with regulatory constraints without revealing any sensitive trajectory information. By leveraging zk-SNARKs, the UAV can generate cryptographic proofs that verify compliance with predefined flight policies while keeping the exact path and location undisclosed. This method mitigates risks associated with real-time tracking, identity exposure, and unauthorised interception, thereby enhancing UAV operational security in adversarial environments. Our proposed solution balances privacy, security, and computational efficiency, making it suitable for resource-constrained UAVs in both civilian and military applications.

Citation: Shayesta Naziri et al. ZAPS: A Zero-Knowledge Proof Protocol for Secure UAV Authentication with Flight Path Privacy. Advances in Knowledge-Based Systems, Data Science, and Cybersecurity.Research 2025;2(2):12.
Page No. 236-259

Keywords: UAV, Authentication, Privacy, ECDH, Zero-Knowledge Proof, Zk-SNARKs.

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are autonomous flying robots that vary in size and capability, from long-range UAVs that can cover hundreds of miles to small, agile drones built for operating in tight spaces [1]. The rapid advancement of drone technologies has paved the way for a new generation of intelligent, autonomous delivery systems. These systems are increasingly being adopted across a wide spectrum of domains including e-commerce, healthcare, agriculture, and defense for their ability to provide fast, efficient, and contactless delivery. As UAV technology becomes more embedded in urban and sensitive operational environments, the need to secure drone communications and preserve user and mission privacy is growing more critical than ever [2, 3].

Among the most pressing concerns in secure drone delivery systems is the flight path privacy problem. Unlike conventional delivery mechanisms, UAVs operate in open-air spaces and are guided by flight plans that often include real time updates, geolocation tracking, and route-specific telemetry. The disclosure of such information whether through intercepted messages, network leaks, or internal compromise can expose sensitive details such as the origin and destination of deliveries, user identities, operational timings, and overall mission intent. This not only endangers user privacy but also invites a range of attacks such as drone hijacking, delivery interception, location profiling, and traffic analysis [4, 5].

In drone delivery ecosystems, multiple entities typically participate in mission coordination, including the user (sender/receiver), the drone, and a central server or ground station. In traditional designs, these entities exchange cryptographic messages for authentication and coordination, but their identities and the drone's trajectory are often either shared or inferable [6]. Even if data encryption is applied, adversaries can perform linkability attacks, timing analysis, or traffic fingerprinting to correlate entities and deduce flight paths. Additionally, adversaries with access to the server or drone can passively harvest metadata to reconstruct the delivery pattern and compromise both identity and location privacy. Insider threats, where entities like the server seek to extract private information, require the protocol to protect privacy even among participants [7, 8]. This sets a high bar for privacy-preserving design, where even legitimate participants must be unable to link identities to routes or extract location details.

Existing solutions to drone security predominantly focus on securing communication channels using symmetric or public-key encryption, message authentication codes, or lightweight key exchange protocols such as ECDH [9]. However, these methods primarily ensure confidentiality and authenticity without providing guarantees of nondisclosure of the underlying route or unlinkability between communicating parties. Furthermore, most traditional protocols lack mechanisms for verifiable computations without revealing sensitive data, making them unsuitable for adversarial environments where zero-trust principles are required.

To address these limitations, we design a protocol that authenticates participants and secures communication while allowing them to prove correct behavior without revealing private data, such as user identities or the drone's flight path. This motivates the use of zero-knowledge proofs specifically, zk-SNARKs which enable proving knowledge of a statement without exposing the

underlying data [10]. We propose a privacy-preserving drone delivery protocol that integrates Elliptic Curve Diffie-Hellman (ECDH) for efficient key exchange and zk-SNARKs for identity and route privacy. The protocol ensures mutual authentication among the User, Drone, and Server, while preserving critical privacy and security guarantees. Flight Path Privacy: The drone's route is not exposed in plaintext; instead, zk-SNARKs are used to prove route correctness without disclosure.

1. The proposed protocol achieves robust flight path privacy through the integration of cryptographic commitments and zero-knowledge proofs, specifically zk-SNARKs. These cryptographic tools allow the drone to prove the correctness of its route without disclosing any portion of the path in plaintext. To further enhance unlinkability, each communication session is initialized with fresh, randomized parameters, including ephemeral keys and unique proofs, ensuring that interactions remain independent and resistant to correlation.
2. A key strength of the proposed protocol lies in its robust privacy preserving architecture, which is specifically designed to ensure that the identities of all involved entities, namely the user, drone, and server, remain completely hidden throughout the communication process. This anonymity is maintained through a combination of cryptographic techniques: secure communication channels are established using Elliptic Curve Diffie-Hellman (ECDH) based key exchange, followed by the application of symmetric encryption to protect data integrity and confidentiality. Additionally, such as the drone's flight path, the protocol leverages zero-knowledge proofs (zk-SNARKs), enabling the verification of route correctness without revealing the actual path and identity protection, making the protocol highly suitable for scenarios demanding stringent privacy guarantees.

The remainder of this paper is organized as follows: Section 2 presents related work in drone authentication and privacy-preserving communication. Section III outlines the system model, design and goals. Section IV details the construction of our proposed protocol, including the ECDH key agreement and zk-SNARK circuit generation. Section V evaluates the protocol's security, privacy, and performance. Finally, Section VI concludes with future directions.

2. RELATED WORK

Flight path privacy, like location privacy in traditional mobile networks, is a crucial concern in the development of UAV systems[11]. However, as there is a lack of protocols specifically designed to protect flight path privacy. To fill this void, we will assess existing protocols that focus on location and other privacy aspects to understand how they might be adapted or extended to ensure better protection of drone flight paths. Various authentication protocols have been proposed to bolster UAV security. For example, [12], combines non-interactive zero-knowledge proofs (NIZKP) with bilinear mapping to achieve unlinkable user identities and resistance to malleability attacks, while [13], employs distance-bounding protocols and anonymous certificates to verify UAV locations without compromising privacy. The multi-factor authentication schemes proposed in [14], and [15], integrate components such as passwords, biometrics, and smart cards to strengthen identity verification. However, they fall short in addressing essential requirements, including replay attack resistance, credential revocation, and protection of location privacy. The authors in [16], implemented a registration hub-based identity management system within the Industrial IoT context,

which, despite offering structured identity handling, does not mitigate location privacy concerns and introduces additional overhead. Similarly, the authors of work [17], employed mobile edge computing with online-offline signatures in their lightweight UAV authentication protocol, but their design still experiences limitations due to constrained memory and processing power.

Moving beyond traditional encryption schemes, in the proposed protocol [18], applied AI techniques to drone surveillance, allowing for feature recognition and tracking over wireless channels. Still, their model experiences significant latency and does not offer strong security guarantees. Similarly, work in [19] and [20], focused on optimizing drone delivery and route planning, but often at the expense of scalability and privacy protections. To address certificate overhead, in the protocol [21], proposed a certificateless approach adaptable to multiple communication settings, including conditional tracking. Nonetheless, it does not adequately secure location privacy and results in high communication costs in batch operations. The works of authors [22], introduces a lightweight security protocol designed to ensure tamper-resistant data exchanges between UAVs and ground control stations, emphasizing integrity and confidentiality. In [23], proposes a secure communication framework to mitigate threats posed by malicious UAVs, ensuring consensus reliability. The work in [24], employs automated algorithms for mission-specific swarm formation selection, streamlining coordination in dynamic environments. In [25], developed a mutual authentication protocol enabling direct communication between drones and users, eliminating the need for intermediaries. However, their scheme is still prone to impersonation and man in the-middle attacks. The authors in [26], put forward a smart card-based authentication method, which, while reasonably secure, is vulnerable to password guessing and physical tampering. In the study [27], they employed elliptic curve cryptography (ECC) to enhance security, but their design involves considerable computational and storage demands due to complex key handling.

Efforts toward lightweight security mechanisms have also been notable. The authors in [28], and [29], proposed using physically unclonable functions and hash-based key agreements to counter standard attacks. Nevertheless, both fail to address the location privacy issue. ECC-based and multi-factor authentication protocols by the authors of works [30], and [31], offer resilience against multiple threats but introduce performance trade-offs and do not tackle location-based vulnerabilities.

In summary, while many current solutions excel in either security enhancement or performance optimization, few successfully achieve both while ensuring drone flight privacy. This work addresses this shortcoming by presenting a secure, efficient, and flight-privacy-preserving authentication protocol designed IoD applications.

3. PRELIMINARIES

3.1 Zero Knowledge Proof

A zero-knowledge proof (ZKP) is a cryptographic protocol where a prover confirms the truth of a statement to a verifier while ensuring that no other information is leaked[32]. Zero-knowledge proofs fall into two main categories: interactive and non-interactive. In interactive proofs, the prover and verifier communicate multiple times, whereas in non-interactive proofs, the prover provides a

single proof that the verifier can check on its own. If a drone uses an interactive proof with an edge server, it will have to hover longer due to the repeated communication steps, leading to greater energy consumption [33].

3.1.1 Zk-SNARKs

A zero-knowledge succinct non-interactive ar arguments of knowledge (zk-SNARKs) scheme represents a specific form of Zero-Knowledge Proof (ZKP) that enables a prover to convince a verifier of their knowledge of a witness, without disclosing any details about the witness itself. This scheme is defined by four key algorithms: Setup, Keygen, Proof, and Verify [34].

Setup$(1^\lambda) \rightarrow \mathcal{Z}$: This algorithm takes as input a security parameter λ and gives as output a set of public parameters $\mathcal{Z} = \{e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$, where p is a prime number, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map [35], and where $\mathbb{G}_1$ and $\mathbb{G}_2$ are acyclic groups (p-order) with generators g_1 and g_2, respectively.

Keygen$(C) \rightarrow (Pk, Vk)$: The Keygen algorithm takes as an input the arithmetic circuit and uses the public parameters $\mathcal{Z}$ to generate a pair of keys for proving and verifying the statement: (Pk, Vk).

Genproof$(Pk, x, w) \rightarrow \pi$: The Genproof algorithm takes as input the proof key that was generated by the Keygen algorithm, the statement x (input of circuit C) and a secret witness w (auxiliary input of circuit C), and generates a zero-knowledge proof π based on the relation between the circuit C, the statement x and the witness w.

Verproof$(Vk, x, \pi) \rightarrow b$: The Verproof algorithm takes as input the verification key Vk, the statement x, and the proof π and generates as output a binary number based on the proof's π validity. If the proof is valid, then $b = 1$ or else $b = 0$.

The zk-SNARK scheme satisfies the following properties:

Completeness: If a statement is $x \in \mathcal{L}$, and w is a valid witness of x, then the verifier accepts the proof with probability 1.

$$\Pr[\text{Verproof}(Vk, x, \pi) \rightarrow 1 \mid \text{Genproof}(Pk, x, w) \rightarrow \pi] = 1$$

Zero knowledge: Without revealing any information regarding the witness w, the prover can prove to the verifier that the statement x is true. This can be described mathematically as follows: Let $\mathcal{S}$ be a simulator that, given a statement, $x \in \mathcal{L}$ and the Pk can produce a proof that is indistinguishable from a real proof generated by the prover, without knowing the witness w. Then,

$$\{\text{Genproof}(Pk, x, w)\} \approx \{\mathcal{S}(Pk, x)\},$$

where "$\approx$" denotes computational indistinguishability.

Soundness: If a witness w is not valid, then a malicious actor cannot craft a proper proof. If we denote the generated malicious proof with $\tilde{\pi}$, then

$$\Pr[\text{Verproof}(Vk, x, \tilde{\pi}) \rightarrow 1] \leq \epsilon,$$

where ϵ negligible.

3.1.2 Hash function

We use a hash function

$$H : \{0,1\}^* \rightarrow \{0,1\}^{\lambda}$$

where λ is a positive integer determined by the security parameter. This function is assumed to have the following key property:

Collision Resistance: It should be computationally infeasible for an adversary to find two distinct inputs m_1 and m_2 such that

$$H(m_1) = H(m_2).$$

Furthermore, we rely on the fact that if a hash function is collision-resistant, it also provides the related security properties of *one-wayness* (hard to invert) and *second preimage resistance* (hard to find a different input that hashes to the same output).

3.1.3 Elliptic curve Diffie-hellman (ECDH)

Given an elliptic curve E over a finite field $\mathbb{F}_q$, and a base point $G \in E(\mathbb{F}_q)$ of prime order n, two parties A and B wishing to share a secret key K do so as follows:

- A selects a random scalar $a \in \mathbb{Z}_n$ and sends $M_A = aG$ to B.
- B selects a random scalar $b \in \mathbb{Z}_n$ and sends $M_B = bG$ to A.
- A computes $K = aM_B = abG$, and B computes $K = bM_A = abG$.

Both parties arrive at the same shared secret $K = abG$, which can then be used as a symmetric key or further processed using a Key Derivation Function (KDF).

4. SYSTEM AND PROPOSED PROTOCOL

4.1 Main Entities

As illustrated in FIGURE 1. in the proposed drone delivery system, there are three main entities: the User (U), the Drone (D), and the Server (S). Their interactions and roles are depicted in the protocol described above [36, 37].

- User (U): The User/Recipient in a drone delivery system is the central entity that initiates the delivery request, communicates securely with the Server and Drone, and confirms the receipt of the delivered item.
- Drone (D): The Drone in a drone delivery system is a critical component responsible for physically transporting goods, communicating with the Server and User, and ensuring the safe and secure delivery of packages. Its role is essential in facilitating the entire delivery process, from initialization to final acknowledgment.

- Server (S): In a drone delivery system, the Server can also be considered as a Ground Control Station or Sender, which plays a crucial role in managing and coordinating the entire delivery process. This server acts as the central hub for communication, control, and data management. It is responsible for generating and managing cryptographic keys, verifying authentication tokens, and ensuring secure communication between the User, the Drone, and itself. The Ground Station Server calculates optimal flight paths for the Drone, monitors its status in real-time, and adjusts flight plans as needed to address any unexpected situations. Additionally, it stores and manages all relevant data, including user information, drone status, and delivery requests, ensuring efficient order processing and transaction management.

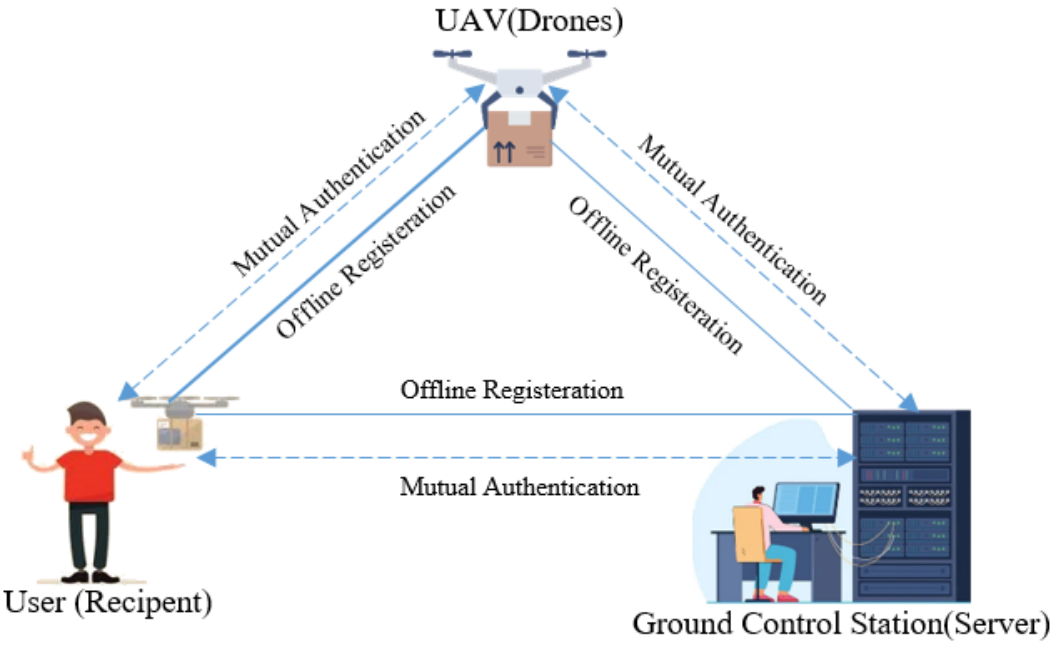

Figure 1: System Model of the ZAPS protocol.

4.2 System Description

The proposed secure drone delivery system consists of three main entities: the Server, the Drone, and the Recipient (User), working together to ensure privacy-preserving and authenticated delivery using elliptic curve cryptography (ECDH) and zk-SNARKs. The zk-SNARK parameters—specifically the proving key P_k and verification key V_k—are generated during a one-time trusted setup phase conducted entirely offline prior to system deployment. This trusted setup is assumed to be carried out in a secure environment, and any toxic waste (intermediate randomness) is securely discarded to prevent compromise. Once generated, P_k and V_k are securely distributed to the corresponding system entities.

As illustrated in FIGURE 2, the process begins when a delivery is initiated, and the Server initializes both the Drone and Recipient by registering their identities and cryptographic credentials. Once

initialization is successful, secure keys are exchanged between the Drone and the Recipient using an ECDH-based key agreement to establish a shared session key for encrypted communication.

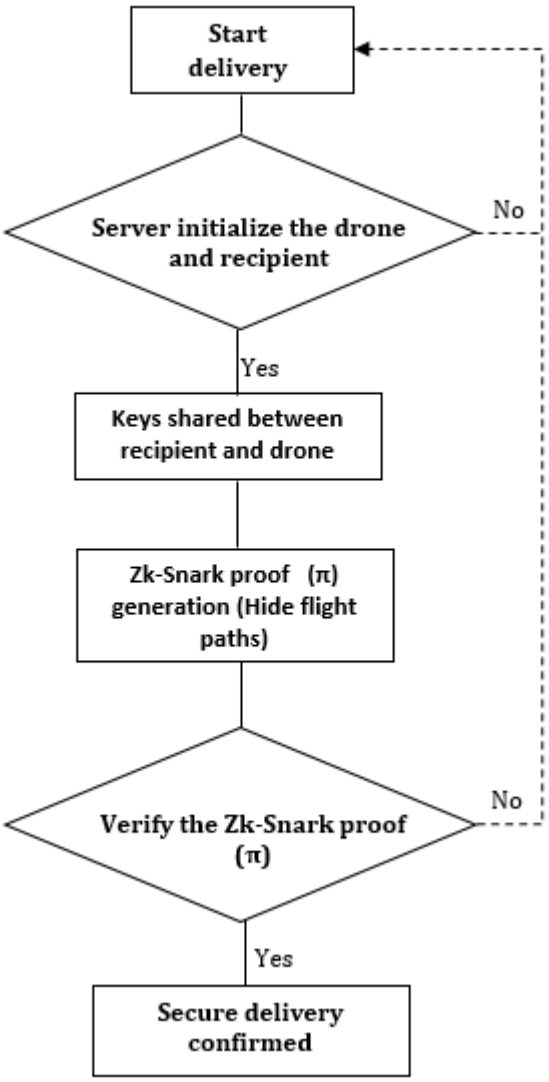

Figure 2: Flowchart of the ZAPS protocol

To ensure privacy of sensitive delivery details, such as the Recipient's identity and the Drone's flight path, the Recipient generates a zk-SNARK proof (π) that validates delivery authorization without revealing any private data. This proof is then verified by the Drone or Server using the public verification key V_k. If the zk-SNARK proof is valid, the Drone proceeds with the delivery. Finally, the Recipient confirms the secure reception of the package using a cryptographically signed acknowledgment.

This end-to-end protocol ensures strong authentication, delivery integrity, anonymity, and flight path confidentiality while operating under clearly defined trusted setup assumptions.

4.3 Proposed Protocol

The protocol employs a four-phase to ensure secure and private operations in drone delivery systems: (1) System Setup Phase, (2) Registration Phase, (3) Initialization and authentication and (4) zk-SNARK proofs generation for flight path privacy. Furthermore, the notations with their significance tabulated in TABLE 1, are utilized in describing as well as analyzing protocol contains the following phases.

Table 1: Notation used in this protocol

Notation	Description
U_u	*u User (Recipient)*
GCS_s^s	*s^{th} Ground Control Station (Server)*
D_d	*d^{th} Drone*
ID_i, PW_i	*Identity, Password*
π	*zk-SNARK proof*
C_i	*Commitment value*
W_i	*Private witness (e.g. flight path)*
r_i	*Random value*
$V_{k,i}$, $P_{k,i}$	*Verification and proving keys for proofs*
V, P	*Private and Public key*
G	*Base point of elliptic curve*
$\mathbb{Z}_p^*$	*Multiplicative group of order $p-1$*
$\mathbb{F}_p$	*Finite field*
$E_p(a,b)$	*Elliptic curve over $\mathbb{F}_p$*
ΔT	*Maximum transmission delay*
CT	*Current timestamp*
Auth	*Verification information*
I	*Hash digest*
SK	*Session key*
SMK	*Symmetric key*
$\tilde{A}$	*The adversary*
$ECDH$	*Elliptic-Curve Diffie–Hellman*
$h(\cdot)$	*One-way hash function*
$\|$	*Concatenation*

4.3.1 Setup phase

In this phase, the Control Authority (CA) initializes system-wide parameters and cryptographic elements. The server or Ground Control Station (GCS) begins by selecting a non-singular elliptic curve $E_p(a,b)$ defined over a prime finite field $\mathbb{Z}_p$, where p is a large prime satisfying the condition $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. A base point G on the curve is chosen with a prime order n, such that $n \cdot G = O$, where O denotes the point at infinity. The server selects a private key $V_s \in \mathbb{Z}_p$ and computes its corresponding public key $P_s = V_s \cdot G$. A collision-resistant cryptographic hash function

$H(\cdot)$, such as SHA-256, is also chosen. The public system parameters $\{E_p(a,b), p, G, H(\cdot), P_s\}$ are published, while the private key V_s remains confidential and is securely stored by the server.

4.3.2 Registration phase

All participating entities (e.g., User, Drone, and Server) are securely registered by the CA before network deployment. The Control Authority securely registers users and drones with the server in an offline environment. For each user U_i, a unique identity ID_u is assigned.

A private key $V_u \in \mathbb{Z}_p^*$ is generated, and the corresponding public key is computed as $P_u = V_u \cdot G$. A symmetric key $SMK_{u,s} = V_u \cdot P_s$ is derived for secure communication with the server. The information set $\{ID_u, V_u, SMK_{u,s}, H(\cdot)\}$ is securely pre-loaded into the user's device. Similarly, for each drone D_i, a unique identity ID_d is assigned, and a private key $V_d \in \mathbb{Z}_p^*$ is selected. The corresponding public key is computed as $P_d = V_d \cdot G$. The server S_i independently generates its own private key $V_s \in \mathbb{Z}_p^*$, computes the public key $P_s = V_s \cdot G$, and publishes P_s as part of the public parameters.

4.3.3 Initialisation and authentication phase

The initialization phase of our proposed protocol follows a similar approach to that described in [38], enhancing it to meet the unique security and privacy demands of our protocol.

In this phase, all registered entities mutually authenticate each other using ECDH-based exchange necessary cryptographic elements to enable zk-SNARK-based privacy-preserving operations. The user begins by selecting a random nonce $r_1 \in \mathbb{Z}_p^*$ and a secret V_u. The user computes their public key $P_u = V_u \cdot G$, an ephemeral value $R_u = r_1 \cdot P_s$, and the symmetric key $\mathsf{SMK}_{u,s} = V_u \cdot V_s \cdot G$. Using these values, the user calculates a hashed identity commitment $I_u = H(\mathsf{PWD}_u \,\|\, \mathsf{ID}_u \,\|\, \mathsf{SMK}_{u,s})$, and constructs the authentication token $\mathsf{Auth}_u = P_u \,\|\, \mathsf{SMK}_{u,s}(P_u, \mathsf{RID}_u, R_u, I_u)$. The user then sends message $\mathsf{Msg}_1 = \{P_u, \mathsf{RID}_u, I_u\}$ to the server.

Meanwhile, the drone chooses a random nonce $r_2 \in \mathbb{Z}_p^*$ and a secret value V_d. It computes its public key $P_d = V_d \cdot G$, the ephemeral value $R_d = r_2 \cdot P_s$, and the symmetric key $\mathsf{SMK}_{d,s} = V_d \cdot V_s \cdot G$. The drone then computes $I_d = H(\mathsf{ID}_d \,\|\, \mathsf{SMK}_{d,s} \,\|\, R_d)$, and forms its authentication message $\mathsf{Auth}_d = P_d \,\|\, \mathsf{SMK}_d(P_d, \mathsf{DID}_d, R_d, I_d)$. The drone sends $\mathsf{Msg}_2 = \{P_d, \mathsf{DID}_d, I_d, \mathsf{Auth}_d\}$ to the server.

Upon receiving both messages, the server proceeds to verify Auth_u and Auth_d, then computes the authentication hashes $I_{s,u} = H(\mathsf{ID}_s \,\|\, \mathsf{ID}_u \,\|\, R_u \,\|\, P_s \,\|\, \mathsf{SK}_{s,u} \,\|\, R_d \,\|\, P_u)$ and $I_{s,d} = H(\mathsf{ID}_s \,\|\, \mathsf{ID}_d \,\|\, R_d \,\|\, P_u \,\|\, \mathsf{SK}_{s,d} \,\|\, R_u \,\|\, P_s)$, generating response tokens $\mathsf{Auth}_{s,u}$ and $\mathsf{Auth}_{s,d}$. The server then replies with $\mathsf{Msg}_3 = \{\mathsf{SID}_s, \mathsf{UID}_u, R_s, \mathsf{Auth}_{s,u}\}$ to the user and $\mathsf{Msg}_4 = \{\mathsf{SID}_s, \mathsf{DID}_d, R_s, \mathsf{Auth}_{s,d}\}$ to the drone.

In the final key derivation step, the drone calculates the session key as $\mathsf{SK}_{d,u,s} = r_2 \cdot r_1 \cdot V_s \cdot V_u \cdot G$, while the user computes $\mathsf{SK}_{u,d,s} = r_1 \cdot r_2 \cdot V_s \cdot V_d \cdot G$. Since the computations are equivalent, both parties derive the same mutual session key for secure communication. The specific steps are illustrated in FIGURE 3.

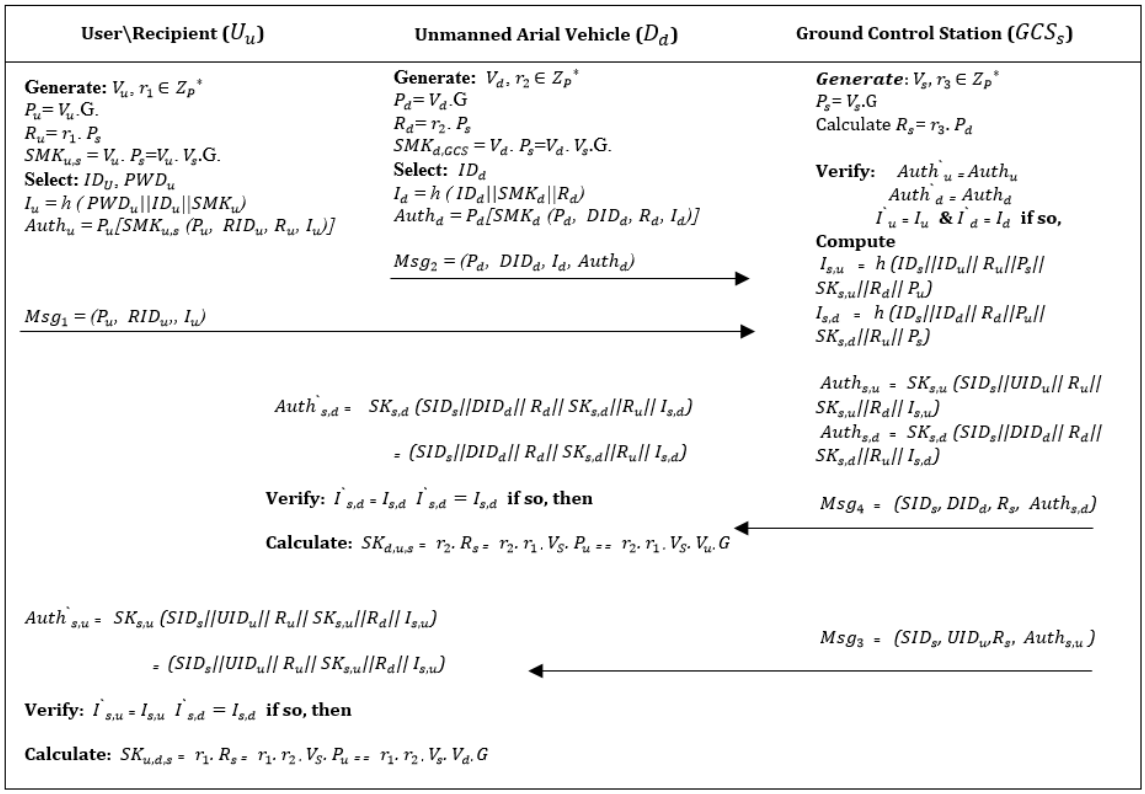

Figure 3: Initialisation and authentication phase of the ZAPS protocol.

4.3.4 zk-SNARK proofs generation phase for flight path privacy

Once mutual authentication is complete, this phase enables entities (e.g., Drone) to prove knowledge of sensitive flight path data without revealing the actual path using zk-SNARKs. As steps are illustrated in FIGURE 4, the user generates a random nonce $r_4 \in \mathbb{Z}_P^*$, a timestamp T_1, and constructs a zk-SNARK circuit to encode their private flight path w_u (treated as a secret witness). Using a Pedersen commitment scheme, the user computes $C_u = r_4 \cdot G + H(w_u \parallel r_4) \cdot G$, where G is the elliptic curve base point. Leveraging tools like Circom and SnarkJS, the user generates a zero-knowledge proof $\pi_u = \text{GenProof}(C_u, P_{k,u}; w_u, r_4, V_u)$ based on precomputed trusted setup parameters. The proof π_u, along with public inputs, is authenticated via $Auth_u = SK_{u,s} \cdot (P_u, UID_u, R_u, I_u)$, where $I_u = H(\pi_u \parallel C_u \parallel SK_u \parallel SRK_u \parallel T_1 \parallel r_4)$. The user then dispatches $Msg_5 = (P_u, RID_u, I_u, \pi_u, Auth_u, T_1)$ to the server.

The *Server Action Phase* begins by verifying the freshness of T_1 using $|T_2 - T_1| < \Delta T$ to thwart replay attacks. The server validates π_u via $\text{VerProof}(V_{k,u}, C_u, P_u, \pi_u)$, generates its own nonce $r_5 \in \mathbb{Z}_P^*$, and computes a commitment C_s with proof π_s. It derives $I_s = H(\pi_s \parallel Msg_5 \parallel r_5)$ and signs $Auth_s = SK_{s,d} \cdot (P_d, SID_s, R_s, I_s)$, transmitting $Msg_6 = (Auth_s, P_s, I_s, \pi_s)$ to the drone.

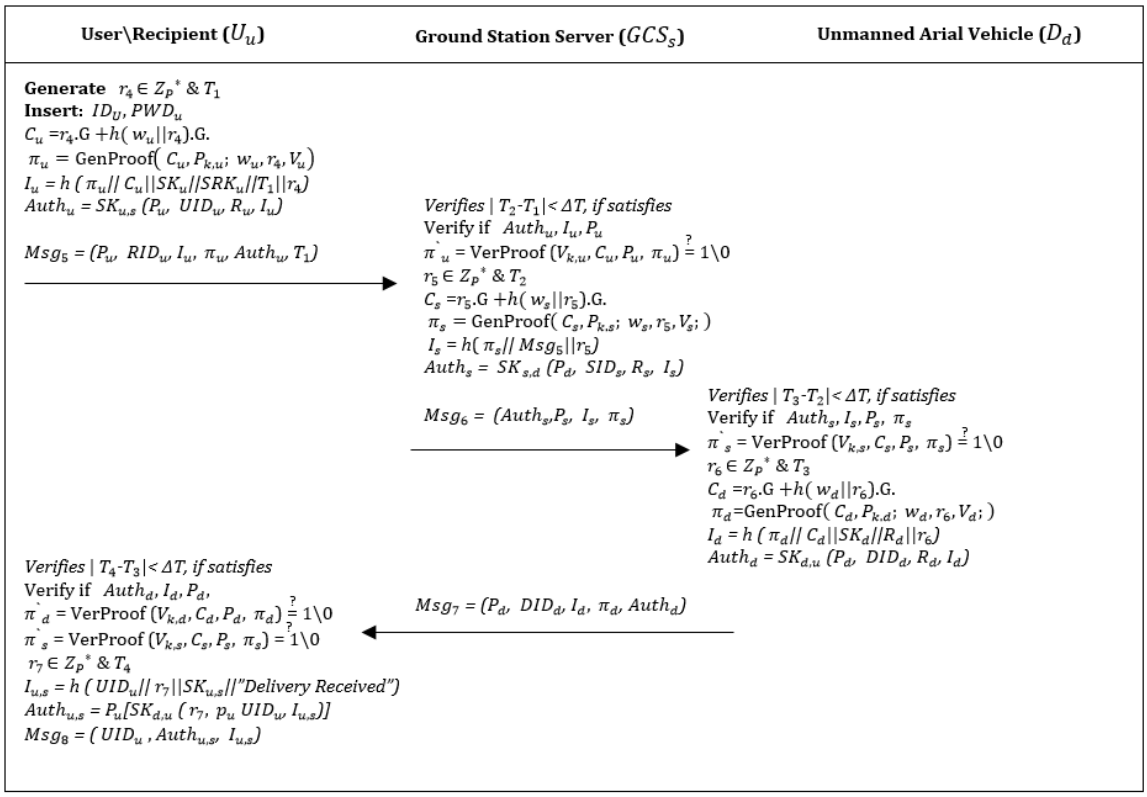

Figure 4: Zk-SNARK Proofs Generation Phase for Flight Path Privacy for the ZAPS protocol.

In the *Drone Action Phase*, the drone verifies the timestamp T_2 against $|T_3 - T_2| < \Delta T$, checks the server's proof π_s, and generates a nonce $r_6 \in \mathbb{Z}_P^*$. It computes its commitment C_d, proof π_d, and authentication token $Auth_d = SK_{d,u} \cdot (P_d, DID_d, R_d, I_d)$, where $I_d = H(\pi_d \parallel C_d \parallel SK_d \parallel R_d \parallel r_6)$. The drone sends $Msg_7 = (P_d, DID_d, I_d, \pi_d, Auth_d)$ to the user.

Finally, during *User Confirmation*, the user verifies $|T_4 - T_3| < \Delta T$, validates π_d and π_s, and confirms delivery by generating an acknowledgment token $Auth_{u,s}$ to complete the protocol.

5. ANALYSIS

We present a detailed examination of the privacy and security properties of the proposed scheme in this section. Additionally, we have conducted a thorough examination of the correctness and other security-related concerns of the proposed scheme.

5.1 Correctness Analysis

5.1.1 Correctness of initialization phase

In this phase, a session key correctly derived by the User U, *Drone* D, *and Server* S *using ECDH. Let* G be the generator of the elliptic curve group G, and let $V_u, V_d, V_s \in \mathbb{Z}_p^*$ be the long-term private keys of the User, Drone, and Server respectively. Their corresponding public keys are defined as $P_u = V_u \cdot G$, $P_d = V_d \cdot G$, and $P_s = V_s \cdot G$. Let $r_1, r_2, r_3 \in \mathbb{Z}_p^*$ be fresh nonces used during the ECDH protocol. The ephemeral key from the User to the Server is $R_1 = r_1 \cdot P_s = r_1 V_s G$, from the Drone to the Server is $R_2 = r_2 \cdot P_s = r_2 V_s G$, and from the Server to the Drone is $R_s = r_3 \cdot P_d = r_3 V_d G$. The session key at the User side is computed as $SK_u = r_1 \cdot R_2 = r_1 \cdot (r_2 V_s G) = r_1 r_2 V_s G$, and at the Drone side as $SK_d = r_2 \cdot R_1 = r_2 \cdot (r_1 V_s G) = r_2 r_1 V_s G = r_1 r_2 V_s G$. Similarly, the session key at the Server side (using both nonces) is $SK_s = V_s \cdot r_1 r_2 G = r_1 r_2 V_s G$. Therefore, $SK_u = SK_d = SK_s = r_1 r_2 V_s G$. The correctness of the Initialization Phase has been proven.

5.1.2 Correctness of zk-SNARK proof generation phase

This protocol uses zk-SNARKs to prove a secret such as a flight path is valid without revealing it. We will break down the process step by step, covering the logic, math, and meaning behind the proof's correctness. In this phase, the goal is to show that the zk-SNARK proof verifies that the committed values correspond to valid private data without revealing it. Let:

- w_i: private witness (e.g., secret flight path, credentials)
- r_i: random nonce for hiding the witness
- $C_i = r_i \cdot G + H(w_i \parallel r_i) \cdot G$: Pedersen-style commitment
- $\pi_i = \text{Prove}(C_i, w_i, r_i, P_i)$: zk-SNARK proof generated by prover i
- $V_{k,i}$: verification key for π_i

The verifier checks if the zk-SNARK proof is valid using a pairing based equation:

$$\text{Verify}(vk_i, C_i, \pi_i) \Rightarrow e(VK + \pi_A, \pi_B) = e(\pi_H, vk_z) \cdot e(\pi_C, h) \tag{1}$$

which ensures three critical things simultaneously: that the proof π was generated correctly, that the values committed in C_i are consistent with the private witness w_i, and that all computations align with the structure defined during the trusted setup phase.

The verification key VK is constructed as

$$VK = A_0(\tau) \cdot \rho_A \cdot G + \sum_{i=1}^{n} s_i A_i(\tau) \cdot \rho_A \cdot G \tag{2}$$

where it aggregates polynomial evaluations at a secret toxic point τ scaled by ρ_A. The proof components π_A, π_B, π_C, and π_H satisfy the equation

$$e\big(A(\tau) \cdot \rho_A \cdot G, B(\tau) \cdot \rho_B \cdot h\big) = e(\pi_H, vk_z) \cdot e(\pi_C, h) \tag{3}$$

which ties the prover's computation to the verifier's expectation.

Leveraging bilinearity, the equation simplifies as

$$e(G,h)^{(H(\tau)Z(\tau)+C(\tau))\cdot\rho_A\rho_B} = e(G,h)^{H(\tau)Z(\tau)\cdot\rho_A\rho_B} \cdot e(G,h)^{C(\tau)\cdot\rho_A\rho_B} \tag{4}$$

breaking down the computation and commitment into separable verifiable components. This culminates in the equality

$$e(\pi_H, vk_z) \cdot e(\pi_C, h) = e(VK + \pi_A, \pi_B) \tag{5}$$

$$\Rightarrow e(\pi_H, vk_z) \cdot e(\pi_C, h) = e(VK + \pi_A, \pi_B) \tag{6}$$

confirming that the left and right sides of the pairing equation match, thus validating the proof.

In summary, the prover commits to a private flight path and generates a zk-SNARK proof attesting that they know a valid witness for the commitment C_i; the verifier, using the trusted setup's verification key and the proof components, confirms correctness through bilinear pairings, ensuring both privacy and integrity of the hidden path.

5.2 SECURITY ANALYSIS

This section provides an in-depth security evaluation of the proposed protocol, covering essential security goals such as confidentiality, authentication, integrity, anonymity, non-repudiation, resilience against various attacks, and time synchronization.

5.2.1 Confidentiality

Confidentiality is achieved through ECDH for secure key exchange. Each participant, the user, the drone and the server, uses its own private key (e.g. r_1, r_2, r_3) within an elliptical curve group to derive a common secret. This ensures that only legitimate entities can access the shared key, keeping the communication secure from eavesdroppers. Even if the communication is intercepted, the shared secret remains undiscoverable without access to the private keys. Additionally, zk-SNARKs protect sensitive details, such as the drone's flight path, by allowing proof of compliance without revealing the actual path. This mechanism upholds flight and location privacy throughout the protocol's execution.

5.2.2 Authentication

Authentication is guaranteed through mutual identity verification among all entities involved, User, Drone, and Server. The protocol uses elliptic curve cryptography and digital signatures for this

purpose. zk-SNARK-based proofs (e.g., Auth_u, Auth_d, Auth_s) confirm each party's knowledge of secret credentials (e.g., I_u, I_d, I_s) without exposing those credentials. Moreover, hashed identifiers (e.g., $H(PWD_u \parallel ID_u)$) conceal user identities during communication, making it infeasible for attackers to deduce real identities from the messages.

5.2.3 Integrity

Integrity of data is preserved through robust cryptographic techniques. Every message exchanged (such as $\text{Msg}_1, \text{Msg}_2, \text{Msg}_3$) contains hash values and authentication tags derived from earlier information, ensuring that any modification is detectable. zk-SNARKs (e.g., π_u, π_d, π_s) appended to the messages validate their authenticity and confirm that the sender has not been impersonated or the content tampered with. The Server also performs verification of these proofs, assuring that messages from both the Drone and User remain intact and authentic.

5.2.4 Anonymity

Anonymity is maintained through a combination of hashed identifiers and zero-knowledge proofs. zk-SNARKs allow each participant to verify knowledge of sensitive data (such as identity or flight path) without exposing it, ensuring complete identity protection. Hashing techniques like $H(PWD_u \parallel ID_u)$ shield real identities during transmission. Furthermore, the protocol safeguards location privacy by hiding the drone's actual travel path using zk-SNARK encryption. As a result, no party including potentially untrusted ones can infer where the drone is or has been during its operation.

5.2.5 Non-repudiation

Non-repudiation is enforced by ensuring that every action taken during the protocol is cryptographically provable. Each party retains evidence of their participation in the form of zk-SNARKs and authenticated message logs (e.g., Msg_5 through Msg_8). These cryptographic artifacts serve as undeniable records, enabling any involved entity to demonstrate that a specific message was sent or received at a certain time, effectively preventing denial of involvement.

5.2.6 Attack resistance

The proposed protocol incorporates a variety of defense mechanisms to resist known attack vectors in secure communication environments.

- Resistance to Man-in-the-Middle (MITM) Attacks: The protocol employs ECDH for secure key agreement, where shared secrets (e.g., $SK_{u,s}$, $SK_{s,d}$) require both parties' private keys. Intercepted public keys alone are insufficient for computation, making MITM decryption or forgery infeasible. Additionally, zk-SNARK proofs authenticate entities without exposing secrets, preventing impersonation or proof forgery.

- Replay Attack Mitigation: Replay attacks are prevented through the use of secure timestamps T_1, T_2, T_3, T_4 embedded within each message. Each entity verifies that the received timestamp is within an acceptable time window, e.g., $|T_2 - T_1| < \Delta T$. This temporal validation ensures that stale or previously valid messages cannot be resent to disrupt the protocol or gain unauthorized access.
- Resistance to impersonation attacks : The proposed protocol resists impersonation attacks using mutual authentication with tokens such as $Auth_u$, $Auth_{s,u}$, $Auth_{s,d}$, $Auth_d$, and $Auth_{d,u}$. These tokens are computed from ECDH-based session keys and bound to identities, nonces, and timestamps. Since only legitimate entities possess the private keys and session-specific secrets, attackers cannot generate valid tokens, effectively preventing impersonation.
- Resistance to Collusion Attack: The protocol thwarts collusion by enforcing independently generated secrets for users (V_u), drones (V_d), and servers (V_s), along with ephemeral nonces (r_1, r_2, r_3). Since these values are never shared or reused, colluding parties cannot combine partial knowledge to reconstruct session keys (e.g., $SK_{u,s} = V_u \cdot V_s \cdot G$). zk-SNARKs ($\pi_u, \pi_d, \pi_s$) further protect privacy by verifying authentication and session setup without exposing sensitive inputs (PWD_u, ID_d, w_s). Thus, even a drone–server collusion cannot recover V_u or forge user-specific proofs derived from $I_u = h(PWD_u \| ID_u \| SMK_u)$. This guarantees strong resistance against collusion during initialization.
- Resistance to Key Compromise Impersonation (KCI): In many authentication protocols, compromising one party's key allows an adversary to impersonate others to that party. In our protocol, KCI is mitigated by mutual authentication and the fact that proof generation in zk-SNARKs depends on knowledge of multiple secret values. Even with access to one entity's private key, an attacker cannot forge valid proofs or compute session keys without the secrets from the other party.
- Message Tampering Detection: Each message includes a cryptographic hash of prior context and zk-SNARK proof elements (e.g., π_u, π_d, π_s) to ensure that any modification or injection of data is detectable. Tampering with even a single bit of a message will result in failed proof verification or hash mismatches, prompting the receiver to reject the message immediately.

5.3 Theoretical and Experimental Analysis

To evaluate the privacy protection and scalability of the proposed ZAPS protocol, we conducted a series of controlled experiments in a simulated UAV communication environment. These experiments examined the core privacy keys, resistance to trajectory reconstruction, session unlinkability, and potential zk-SNARK information leakage, together with scalability indicators, including CPU power consumption, CPU utilization, and communication overhead, as the UAV network size increased.

All experiments were performed on a Windows 11 machine equipped with an Intel Core i7-12700H CPU, 16 GB RAM, and Python 3.11. The following Python libraries were used: `NumPy`, `Pandas`, `Matplotlib`, `Seaborn`, `scikit-learn`, and `dtaidistance`. Message exchange traces were synthetically generated to emulate UAV–GCS communications. For zk-SNARK proof simulation, pseudo-random proof byte arrays conditioned on the witness were used to enable leakage analysis

without heavy cryptographic back-ends. Real proof integration can be performed using `Circom`/`snarkjs` or `ZoKrates` under the Window Subsystem for Linux (WSL).

Each simulated session consisted of an initialization phase (four fixed-size messages), per-waypoint proof transmissions, and a final termination message. Flight paths were generated with lengths in $\{5, 8, 10, 15\}$ waypoints, and message sizes and inter-arrival times were randomized to reflect realistic wireless conditions.

5.3.1 Aggregate privacy outcomes

The Aggregate Privacy Outcomes (FIGURE 5) present a consolidated evaluation of ZAPS against three representative attack vectors: (1) *Session Clustering Attack* – grouping UAV communication sessions by metadata patterns such as timing and message sizes; (2) *Linkability Attack* – determining whether two different sessions originated from the same UAV; and (3) *Proof Distinguishability Attack* – identifying route-related details from zk-SNARK proof structures or byte distributions.

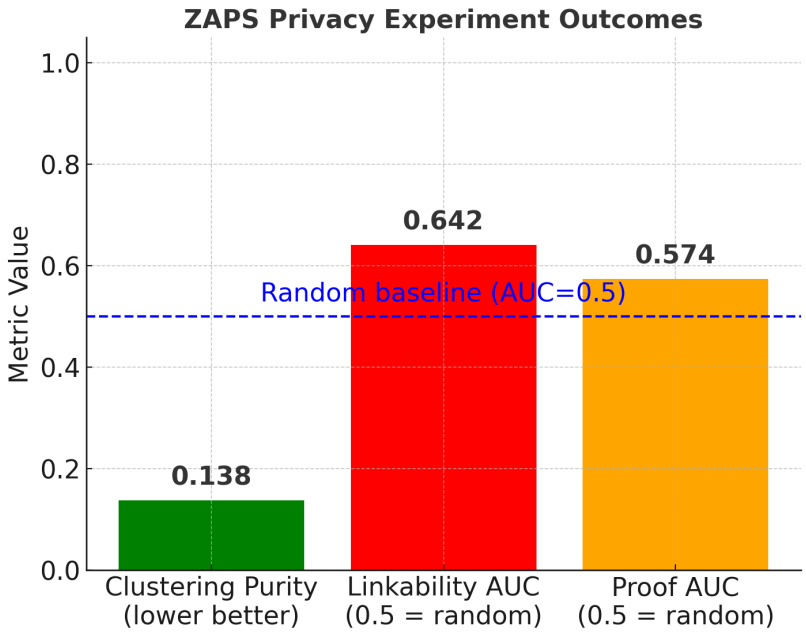

Figure 5: Summary of all privacy metrics (Purity for clustering, AUC for linkability and proof distinguishability) for the ZAPS protocol. *Note:* For AUC-based attacks, 0.5 indicates random guessing (no attack advantage). Values slightly above 0.5 suggest limited leakage, whereas higher deviations from 0.5 indicate stronger attack success.

To simulate these scenarios, synthetic UAV–GCS communication traces were generated with realistic message sizes, timing jitter, and route lengths. The adversary was modeled as a passive global observer with full access to all transmitted metadata but no plaintext content. Metrics used were clustering purity for the session clustering attack, classification accuracy for route inference, and AUC for linkability and proof distinguishability.

In the protected ZAPS configuration, session clustering purity was 0.138, only marginally above the random-mixing baseline (≈ 0). This shows that adversaries could group sessions only slightly better than chance.

The supervised session linkability test achieved an AUC of 0.642, which is only slightly above the 0.5 random baseline, indicating minimal correlation leakage. Similarly, proof distinguishability testing yielded an AUC of 0.574, again close to 0.5. Since $AUC = 0.5$ represents random guessing, values only slightly higher imply that adversaries gain no meaningful advantage from these attacks.

Together, these results confirm that ZAPS strongly resists session linking and proof-based inference, with only negligible statistical leakage.

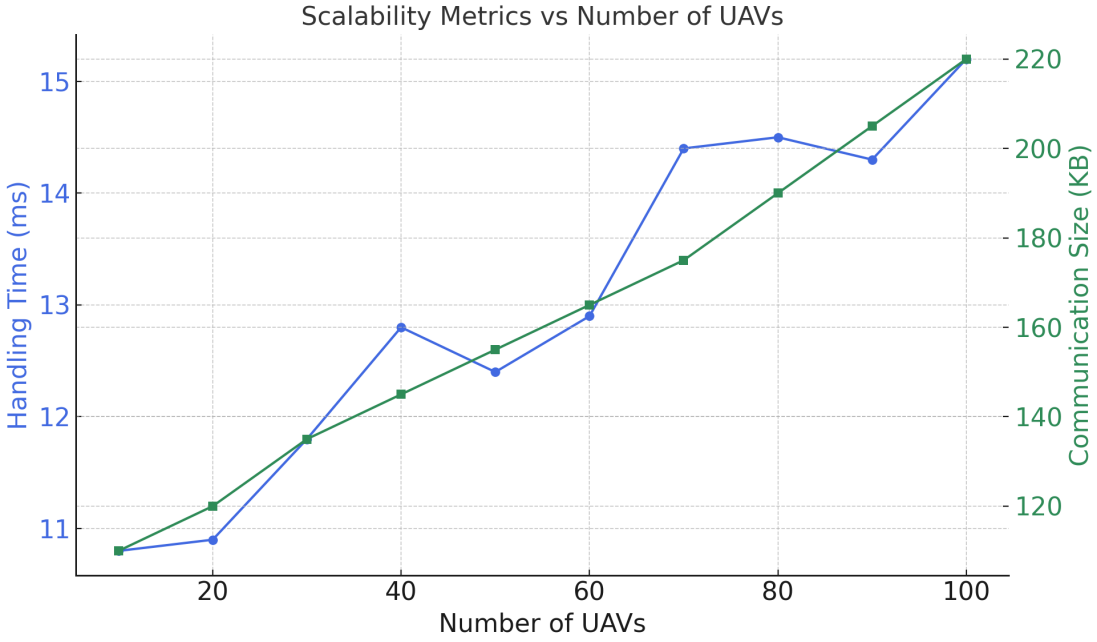

Figure 6: Scalability Metrics for Handling Time and Communication Size vs UAV count in the ZAPS protocol.

5.3.2 Scalability analysis

We simulated network scaling from 10 to 100 UAVs, measuring two key performance indicators: *average handling time* (including proof verification and session processing) and *communication size per UAV*. As shown in FIGURE 6, both metrics exhibit near-linear growth with the number of UAVs, indicating predictable scaling behavior. Handling time increased from approximately 10.8 ms at 10 UAVs to 15.2 ms at 100 UAVs, remaining well within the operational requirements for real-time UAV coordination. Communication overhead grew from 118 KB to 220 KB per UAV over the same range, reflecting the proportional increase in per-UAV proof and message exchanges. This consistent scaling demonstrates that the protocol can accommodate substantial UAV swarm sizes

without exceeding latency or bandwidth thresholds. Furthermore, the low absolute values of both metrics suggest that even deployments with hundreds of UAVs can be supported using commodity hardware and typical wireless data links, making the proposed ZAPS protocol viable for large-scale applications.

5.3.3 Communication overhead

TABLE 2, shows that the communication overhead in the proposed drone delivery protocol is structured across two phases initialization and zk-SNARK Proof Generation accumulating to a total of 1,396 bytes per session. In the initialization phase (480 bytes), four messages are exchanged between the user, the drone, and the server to establish mutual authentication and the setup of sessions. These messages contain components such as elliptic curve public keys (32 bytes), identifiers (16 bytes), random nonces or hashes (32 bytes), and ECHD based authentication tokens (64 bytes), aligning with the assumptions of using secp256k1 for elliptic curve cryptography and SHA-256 for hashing. The zk-SNARK Proof Phase (916 bytes) transmits four additional messages, each embedding Groth16 zk-SNARK proofs (128 bytes), elliptic curve points, authentication tokens, and timestamps (4 bytes), enabling private yet verifiable proof of identity and flight path integrity. This carefully optimized structure balances strong cryptographic guarantees, such as anonymity, freshness, and zero-knowledge flight validation, against minimal communication overhead, making it suitable for privacy-sensitive and resource-constrained drone delivery applications.

Table 2: Communication Overhead

Message	**Components**	**Size (Bytes)**
	1. Initialization Phase	
Msg_1	P_u, RID_u, I_u, $Auth_u$	32 + 144 + 128
Msg_2	$Auth_S$, P_S, I_S, $Auth_{S,u}$	16 + 162 + 62
Msg_3	SID_S, UID_d, R_S, $Auth_{S,d}$	16 + 162 + 32
	Initialization Total:	80 + 144 + 128
	2. zk-SNARK Proof Phase	
Msg_5	P_u, RID_u, I_u, π_u, $Auth_u$	32 + 16 + 32 + 128
Msg_6	$Auth_S$, P_S, I_S, π_S	64 + 32 + 32 + 128
Msg_7	P_d, DID_d, I_d, π_d, $Auth_d$	32 + 16 + 32 + 128
Msg_8	UID_v, $Auth_{u,S}$, $I_{u,S}$	16 + 64 + 32
	zk-SNARK Phase Total:	276 + 256 + 272
	3. Total Communication Overhead	
Total = 480 (Initialization) +916 (zk-SNARK) = 1,396 bytes		

5.3.4 CPU power consumption over time

We adopt the profiles from [39] as a basis for estimating CPU power consumption and CPU utilization over time.These works provide in-depth evaluations of CPU power consumption and processing demands in low-resource environments such as drones and embedded systems.The most power-

intensive part of the protocol is the zk-SNARK proof generation phase performed by the User, Server, and Drone, which typically consumes between 4 and 4.2 watts due to the heavy cryptographic computations involved in generating zero-knowledge proofs. Verification steps executed by the Server, Drone, and User also demand considerable power, averaging around 3 watts. As shown in FIGURE 7. in contrast, other operations such as ECHD based key generation and final authentication are relatively lightweight, consuming only about 1.2 to 1.5 watts. When targeting constrained environments like drones, it may be beneficial to offload proof generation tasks or explore lightweight zk-SNARK variants to reduce power strain. From a comparative standpoint, the 3-watt verification steps represent approximately 5% of the mean power consumption observed in typical performance graphs, while the lighter operations account for just 2–3%. Overall, the power consumption profile of this zk-SNARK-based protocol is highly efficient, especially when contrasted with more resource demanding tasks, making it a viable option for privacy preserving drone applications and other energy sensitive platforms.

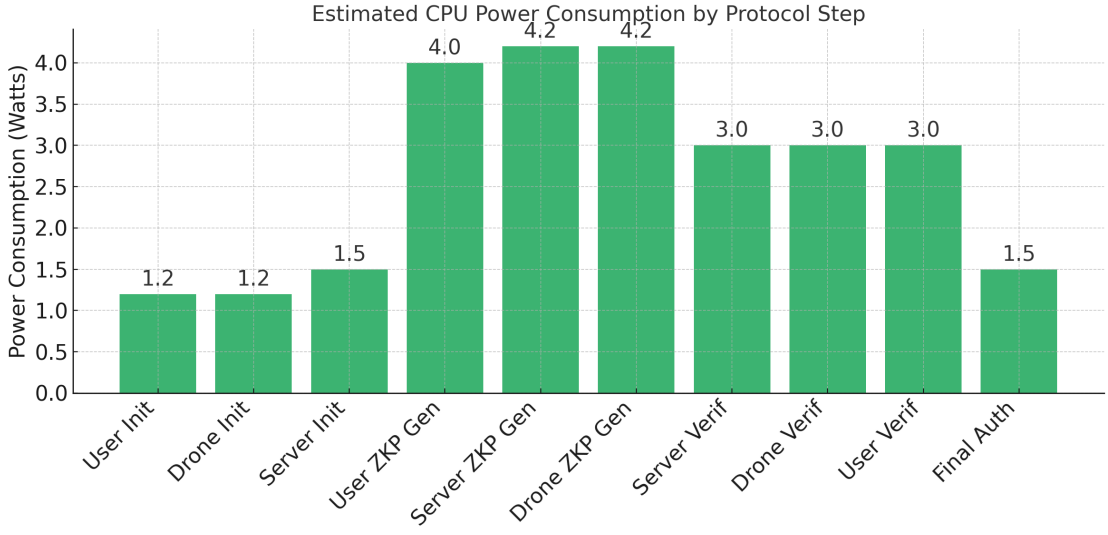

Figure 7: CPU power consumption over time for each step of ZAPS protocol.

5.3.5 CPU utilization over time

The FIGURE 8. illustrates CPU utilization Over Time for zk-SNARKs protocol operations, with the x axis representing time in arbitrary units corresponding to different protocol phases and the y axis showing relative CPU usage intensity. A red line with circular markers plots utilization values across time: starting at 0.8 (Time 0), peaking at 1.0 (Time 1), and then gradually declining to 0.3 by Time 7. This trend suggests that the initial phases, likely involving setup or proof generation, are computationally intensive, while subsequent steps such as verification are less demanding. The steady decline in CPU usage reflects a shift from heavy cryptographic operations to lightweight final steps like authentication. Overall, the plot effectively highlights how zk-SNARKs protocols impose high CPU demands early on, followed by a tapering load as the protocol progresses.

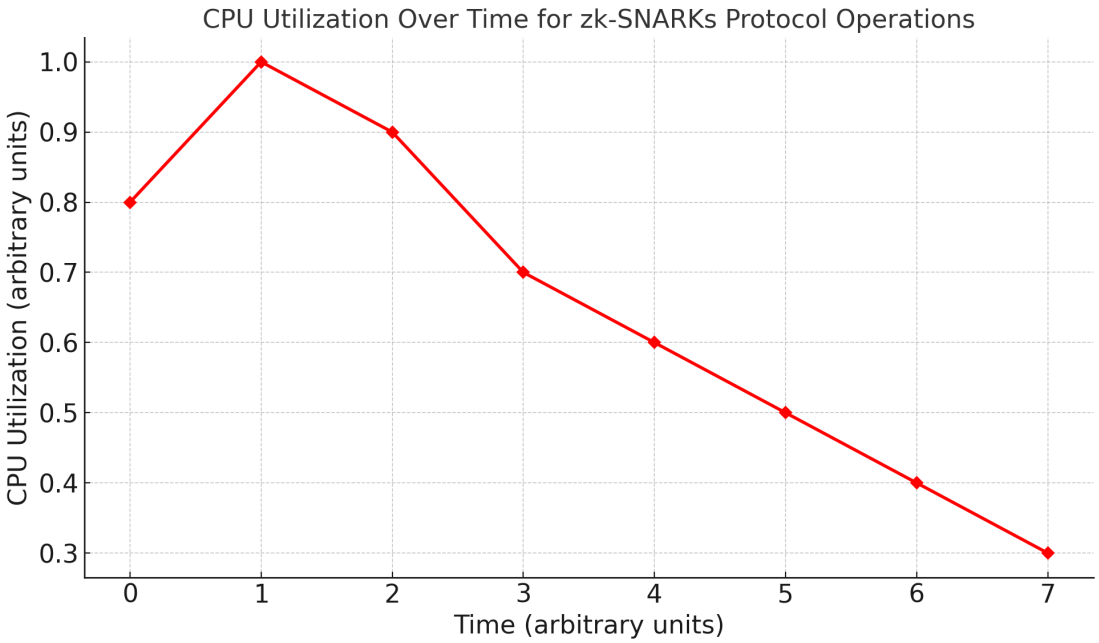

Figure 8: CPU utilization over time for ZAPS protocol.

6. CONCLUSION

This work presented ZAPS, a zero-knowledge proof-based authentication protocol for UAV systems designed to ensure flight path privacy while maintaining robust security against active and passive adversaries. By integrating zk-SNARKs into the UAV–GCS communication process, ZAPS enables mutual authentication and trajectory validation without disclosing sensitive flight path information. Theoretical analysis confirmed the protocol's resilience against common threats, including replay, impersonation, and man-in-the-middle attacks. Experimental evaluation demonstrated strong privacy guarantees, with low clustering purity (0.138) for session unlinkability, minimal proof distinguishability (AUC 0.574), and only marginal linkability leakage (AUC 0.642). Scalability tests showed that ZAPS can support up to 100 UAVs with acceptable computational (15.2 ms average handling time) and communication ($<$ 220 KB per UAV) overhead. Overall, ZAPS delivers a privacy-preserving, scalable, and secure UAV authentication framework, making it a strong candidate for deployment in military, commercial, and civilian drone applications. Future work will focus on deploying ZAPS in real UAV hardware and extending its applicability to diverse domain-specific and cross-domain UAV operations, where location privacy, operational secrecy, and mission-specific confidentiality remain critical.

References

[1] Hassanalian M, Abdelkefi A. Classifications Applications and Design Challenges of Drones: A Review. Prog Aerosp Sci. 2017;91:99-131.

[2] Yang W, Wang S, Yin X, Wang X, Hu J. A Review on Security Issues and Solutions of the Internet of Drones. IEEE Open J Comput Soc. IEEE. 2022;3:96-110.10.1109/OJCS.2022.3183003.

[3] Tanveer M, Alasmary H, Kumar N, Nayak A. Saaf-Iod: Secure and Anonymous Authentication Framework for the Internet of Drones. IEEE Trans Veh Technol. IEEE. 2024;73:232-244.

[4] Mahmood K, Ghaffar Z, Nautiyal L, Akram MW, Kumar Das A, et al. A Privacy-Preserving Access Control Protocol for Consumer Flying Vehicles in Smart City Applications. IEEE Internet Things J. IEEE. 2025;12:978-985.

[5] Zhaolu T, Wan Z, Wang H. Division of Regulatory Power: Collaborative Regulation for Privacy-Preserving Blockchains. IEEE Trans Inf Forensics Sec. 2024;19:2533-2548.

[6] Cho K, Cho M, Jeon J. Fly a Drone Safely: Evaluation of an Embodied Egocentric Drone Controller Interface. Interact Comput. 2016;29:345-354.

[7] Huang C, Ming Z, Huang H. Drone Stations-Aided Beyond-Battery-Lifetime Flight Planning for Parcel Delivery. IEEE Trans Autom Sci Eng. 2023;20:2294-2304.

[8] Miao Y, Hwang K, Wu D, Hao Y, Chen M. Drone Swarm Path Planning for Mobile Edge Computing in Industrial Internet of Things. IEEE Trans Ind Inform. 2023;19:6836-6848.

[9] Xia T, Wang M, He J, Ni L, Yang G. A AAV Swarm Authentication and Key Agreement Scheme Based on Latin Square Design. IEEE Wirel Commun Lett. 2025;14:581-585.

[10] Fiege U, Fiat A, Shamir A. Zero Knowledge Proofs of Identity. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. ACMDL. 1987:210-217.

[11] Ma B, Wang X, Lin X, Jiang Y, Sun C, Wang Z et al. Location Privacy Threats and Protections in Future 6G Vehicular Networks: A Comprehensive Review. arXiv preprint: https://arxiv.org/pdf/2305.04503.

[12] Andola N, Gahlot R, Yadav VK, Venkatesan S, Verma S. SpyChain: AX Lightweight Blockchain for Authentication and Anonymous Authorization in IoD. Wirel Pers Commun. Research Gate. 2021;119:1-20.

[13] Pan H, Wang Y, Wang W, Cao P, Ye F, et al. Privacypreserving Location Authentication for Low-Altitude Uavs: A Blockchain-Based Approach. Secur S Afr. 2024;3:2024004.

[14] Tai WL, Chang YF, Li WH. An IOT Notion-Based Authentication and Key Agreement Scheme Ensuring User Anonymity for Heterogeneous Ad Hoc Wireless Sensor Networks. J Inf Secur Appl. 2017;34:133-141.

[15] Wazid M, Das AK, Odelu V, Kumar N, Conti M, et al. Design of Secure User Authenticated Key Management Protocol for Generic IOT Networks. IEEE Internet Things J. IEEE. 2018;5:269-282.

[16] Singh J, Gimekar A, Venkatesan S. An Efficient Lightweight Authentication Scheme for Human-Centered Industrial Internet of Things. Int J Commun Syst. 2019;2:e4189.

[17] Tian Y, Yuan J, Song H. Efficient Privacy-Preserving Authentication Framework for Edge-Assisted Internet of Drones. J Inf Secur Appl. 2019;48:102354.

[18] Yue X, Liu Y, Wang J, Song H, Cao H. Software Defined Radio and Wireless Acoustic Networking for Amateur Drone Surveillance. IEEE Commun Mag. 2018;56:90-97.

[19] Bouman P, Agatz N, Schmidt M. Dynamic Programming Approaches for the Traveling Salesman Problem With Drone. Networks. 2018;72:528-542.

[20] Shavarani SM, Mosallaeipour S, Golabi M, Izbirak. A Congested Capacitated Multi-Level Fuzzy Facility Location Problem: An Efficient Drone Delivery System. Comput Oper Res. 2019;108:57-68.

[21] Won J, Seo SH, Bertino E. Bertino. Certificateless Cryptographic Protocols for Efficient Drone-Based Smart City Applications. IEEE Int.J. 2017;5:3721-3749.

[22] Allouch A, Cheikhrouhou O, Koubâa A, Toumi K, Khalgui M, et al. UTM-chain: Blockchain-Based Secure Unmanned Traffic Management for Internet of Drones. Sensors . 2021;21:3049.

[23] Alsamhi SH, Shvetsov AV, Shvetsova SV, Hawbani A, Guizani M, Alhartomi MA et al. Blockchain-Empowered Security and Energy Efficiency of Drone Swarm Consensus for Environment Exploration. IEEE Trans Green Commun Netw. IEEE. 2023;7:328-338.

[24] Koulianos A, Litke A. Blockchain Technology for Secure Communication and Formation Control in Smart Drone Swarms. Future Internet. 2023;15:344.

[25] Turkanovic M, Brumen B, Holbl M. A Novel User Authentication and Key Agreement Scheme for Heterogeneous Ad Hoc Wireless Sensor Networks Based on the Internet of Things Notion. Ad Hoc Network. 2014;20:96-112.

[26] Amin R, Islam SH, Biswas GP, Khan MK, Leng L, et al. Design of an Anonymity-Preserving Three-Factor Authenticated Key Exchange Protocol for Wireless Sensor Networks. Comput Netw. 2016;101:42-62.

[27] Challa S, Wazid M, Das AK, Kumar N, Reddy AG, et al. Secure Signature-Based Authenticated Key Establishment Scheme for Future Iot Applications. IEEE Int J. 2017;5:3028-3043.

[28] Gope P, Sikdar B. An Efficient Privacy-Preserving Authenticated Key Agreement Scheme for Edge-Assisted Internet of Drones. IEEE Trans Veh Technol. 2020;69:13621-13630.

[29] Zhang Y, He D, Li L, Chen B. A Lightweight Authentication and Key Agreement Scheme for Internet of Drones. Comput Commun. 2020;154:455-464.

[30] Ever YK. A Secure Authentication Scheme Framework for Mobile-Sinks Used in the Internet of Drones Applications. Comput Commun. 2020;155:143-149.

[31] Hussain S, Mahmood K, Khan MK, Chen CM, Alzahrani BA, et al. Designing Secure and Lightweight User Access to Drone for Smart City Surveillance. Comput Stand Interfaces. 2022;80:103566.

[32] Hao M, Shang C, Wang S, Jiang W, Nie J. Uav-Assisted Zero Knowledge Model Proof for Generative AI: A Multi-Agent Deep Reinforcement Learning Approach. IEEE Internet of Things Journal. 2025;12:13441–13454.

[33] Hao M, Shang C, Wang S, Jiang W, Nie J. UAV-Assisted Zero Knowledge Model Proof for Generative AI: A Multi-Agent Deep Reinforcement Learning Approach. IEEE Internet Things J. IEEE. 2025;12:13441-13454.

[34] Parno B, Howell J, Gentry C, Raykova M. Pinocchio: Nearly Practical Veri Able Computation. Commun ACM. 2016;59:103-112.

[35] Miret JM, Sadornil D, Tena JG. Pairing-Based Cryptography on Elliptic Curves. Math Comput Sci. Sep. 2018;12:309-318.

[36] Dissanayaka D, Wanasinghe TR, De Silva O, Jayasiri A, Mann GK. Review of Navigation Methods for UAV-Based Parcel Delivery. IEEE Trans Autom Sci Eng. Jan. 2024;21:1068-1082.

[37] Deng Z, Wu F, Xu Y, Yang D, Xiao L. Energy Minimization for Radio Map-Based Uav Pickup and Delivery Logistics System. IEEE Trans Veh Technol. 2024;73:17893-17898.

[38] Naziri S, Wang X, Yu G, Xu J, Shrestha S, et al. SMAKAP: Secure Mutual Authentication and Key Agreement Protocol for RFID Systems. 17th International Conference on Security of Information and Networks (SIN) Sydney Australia. 2024:1-8.

[39] Koulianos A, Paraskevopoulos P, Litke A, Papadakis NK. Enhancing Unmanned Aerial Vehicle Security: A Zero-Knowledge Proof Approach With Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge for Authentication and Location Proof. Sensors . 2024;24:5838.