Design, Formal Verification, and Evaluation of a Post-Quantum EDHOC Variant for UAV Communications

Rodrigo Duarte de Meneses, Caio Teixeira, Marco Amaral Henriques

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas – Campinas, SP –Brazil

r197962@dac.unicamp.br, caio@dca.fee.unicamp.br, maah@unicamp.br

Abstract. *Unmanned Aerial Vehicle (UAV) swarms require secure and efficient protocols for communication in dynamic and adversarial environments. This paper presents a post-quantum authenticated key exchange protocol for UAV communications, inspired by recent post-quantum extensions of EDHOC. The protocol is modeled in SAPIC$^+$ and verified in the Tamarin Prover under a Dolev–Yao adversary, establishing executability, mutual authentication, key secrecy, and forward secrecy properties. Communication overhead and handshake latency are estimated using standardized parameter sizes and benchmark data for ARM Cortex-M4 and x86 platforms, showing how the choice of post-quantum primitives affects the feasibility of authenticated key exchange in constrained UAV deployments.*

1. Introduction

Unmanned Aerial Vehicle (UAV) swarms are increasingly deployed in applications such as disaster response, cooperative sensing, and search-and-rescue operations, where multiple peers must coordinate through the continuous exchange of control and telemetry data. These systems operate over open and dynamic communication environments, where adversaries may intercept, modify, or inject messages. As a result, establishing secure communication channels is a fundamental requirement for reliable operation.

Authenticated Key Exchange (AKE) protocols provide the cryptographic foundation for this task by enabling two parties to establish a shared secret key over an adversarial channel while ensuring authentication and key secrecy [Boyd and Mathuria 2019]. In UAV settings, AKE protocols must satisfy additional constraints, including low latency, reduced communication overhead, and computational efficiency on resource-constrained platforms. Moreover, the emergence of quantum computing poses a threat to classical public-key cryptography, motivating the adoption of post-quantum cryptographic primitives.

Recent efforts within the Internet Engineering Task Force (IETF) have proposed lightweight AKE protocols such as RFC 9528 [Selander et al. 2024], as well as initial approaches toward quantum-resistant variants [Papon and Onete 2026, Fraile et al. 2026]. However, adapting EDHOC to post-quantum security is not a simple matter of replacing cryptographic primitives. Post-quantum public keys, ciphertexts, and signatures are substantially larger than their classical counterparts, and different primitive choices can lead

This work was supported by Ericsson Telecomunicações Ltda., and by the São Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. GPT-5.6 Sol was used to assist with language revision. All suggestions were reviewed by the authors.

to significant communication and computational overhead. This is particularly relevant for UAV swarms, where nodes operate under strict bandwidth, latency, and energy constraints.

Beyond efficiency, security assurance is also a central concern. Many AKE proposals for UAV communications rely primarily on informal security arguments, which may be insufficient to rule out subtle protocol-level flaws [Blanchet 2016]. This is particularly relevant when adapting protocols to post-quantum primitives, since changing the cryptographic primitives may affect the security assumptions of the original design. Formal verification provides a more rigorous way to state and check security properties, yielding machine-checked proofs that the protocol satisfies its intended security goals.

Our contribution. This work proposes a post-quantum AKE protocol for UAV communications based on a modified post-quantum EDHOC design, and provides a formal security analysis using the TAMARIN Prover. The analysis shows that the modification preserves the intended security properties, establishing executability, mutual authentication, session key secrecy, and forward secrecy under the stated adversary model. We also provide a performance evaluation, estimating communication overhead and handshake latency for different post-quantum cipher suites. The results identify ML-KEM with ML-DSA as the most practical instantiation for the considered UAV communication scenarios.

The remainder of the paper is organized as follows: Section 2 reviews background on UAV communications, post-quantum cryptography, and AKE protocols. Section 3 defines the system and adversary model, and security goals. Section 4 presents the protocol design. Section 5 describes the formal verification and results. Section 6 evaluates the communication overhead and handshake latency. Section 7 concludes the paper.

2. Background

2.1. UAV Communication

UAV systems are composed of multiple aerial nodes that cooperate to accomplish a shared mission under dynamic and often adversarial conditions. Communication underpins this cooperative behavior by enabling nodes to exchange state information, coordinate actions, and maintain a consistent view of the environment. Swarm communication must support both global coordination with a control entity and local interactions among neighboring nodes, often under strict latency and reliability constraints [Aissaoui 2024]. In a typical deployment, communication follows a hybrid pattern – a central control entity, commonly referred to as the Ground Control Station (GCS), is responsible for mission-level orchestration, while individual UAVs engage in direct peer-to-peer exchanges to support distributed sensing, formation control, and collision avoidance. This creates a communication environment with centralized and decentralized interactions, each with distinct performance requirements. The system must operate across infrastructure-assisted links and direct device-to-device channels, adapting to varying connectivity conditions.

2.2. Post-Quantum Cryptography

Post-quantum cryptography (PQC) studies cryptographic schemes intended to remain secure against adversaries equipped with a cryptographically relevant quantum computer. This is particularly relevant for cryptographic protocols, as classical protocols commonly rely on traditional public-key cryptography. These primitives are vulnerable to Shor's algorithm, which enables integer factorization and discrete logarithm computation in

polynomial time, making such protocols unsuitable for long-term post-quantum security. The National Institute of Standards and Technology (NIST) has introduced the first standards for quantum-resistant key encapsulation mechanism (KEM) and digital signatures. FIPS 203 specifies ML-KEM for key encapsulation [NIST 2024b], while FIPS 204 and FIPS 205 specify ML-DSA and SLH-DSA for digital signatures, respectively [NIST 2024a, NIST 2024c]. Additional algorithms remain in the standardization pipeline: Falcon has been selected for future standardization as FN-DSA [NIST 2025b], and HQC has been selected as an additional KEM [NIST 2025a]. These efforts provide a practical basis for replacing classical public-key mechanisms with post-quantum alternatives.

2.3. Authenticated Key Exchange

Authenticated Key Exchange (AKE) protocols enable two parties to establish a shared secret key over an insecure network while simultaneously authenticating each other's identities [Boneh and Shoup 2023]. In adversarial environments, AKE constitutes the fundamental protocol for bootstrapping secure channels, from which higher-level services can be derived. An AKE protocol is executed between an initiator $\mathcal{I}$ and a responder $\mathcal{R}$, each holding long-term credentials and generating ephemeral values during the protocol run. The execution results in a session key k, which is used to protect application-layer communications between $\mathcal{I}$ and $\mathcal{R}$. In addition to public-key primitives, AKE protocols rely on symmetric primitives for key derivation and protected payload encryption. HMAC-based Key Derivation Function (HKDF) is used for key derivation, while Authenticated Encryption with Associated Data (AEAD) protects encrypted protocol payloads.

Recent work on AKE protocols for UAV networks has followed multiple directions. In [Wu et al. 2025], the authors emphasize lightweight deployment in FANETs through the use of physical unclonable functions, but do not primarily address post-quantum security. In [Xia et al. 2024], the authors move toward quantum-resistant UAV authentication but retain elliptic curve components such as ECDH, thereby not yielding a fully post-quantum AKE design. More recent proposals, such as [He et al. 2025], adopt post-quantum algorithms and dynamic group-management mechanisms, but their security analysis relies on informal arguments. None of these proposals provides a formal security verification of the AKE core, which motivates the study presented in this work.

3. System and Adversary Model

3.1. System Model

We consider a distributed UAV swarm composed of multiple aerial nodes and a GCS operating in a dynamic communication environment. Communication occurs over two types of links: (i) UAV–GCS links, used for centralized coordination and mission control, and (ii) UAV–UAV links, used for peer-to-peer interaction, such as formation control and collision avoidance. Figure 1 shows an illustration of this communication model.

Each party is provisioned with long-term credentials and generates ephemeral values during protocol execution. In particular, every participant holds a long-term signature key pair $(sk^{\mathsf{sig}}, pk^{\mathsf{sig}})$, which is used for authentication, and generates ephemeral key material through a post-quantum KEM during each session. Session keys are derived from the shared secret established via KEM and are used to protect subsequent communications. The protocol assumes a certificate-based credential model, in which each party holds a

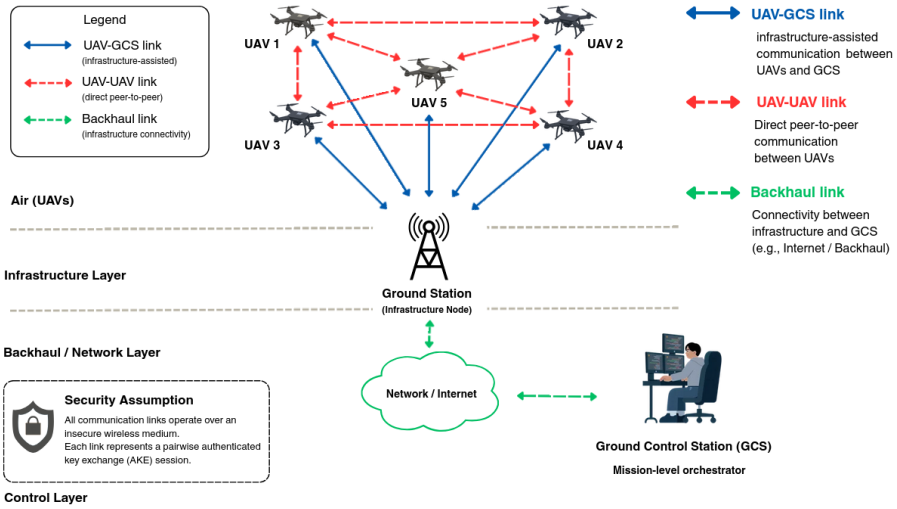

Figure 1. Overview of the communication model considered for this work.

certificate issued by a trusted Certificate Authority (CA), binding its identity to its public signature key. During protocol execution, peers exchange certificates and verify them using a locally provisioned, authenticated copy of the CA public key pk_{CA}. This model enables scalable and dynamic deployments, where nodes may join or interact without prior pairwise provisioning. All honest parties are assumed to be uniquely represented by an identifier bound to their corresponding public keys, to follow the protocol specification and to maintain local state for ongoing sessions, including transcript values and derived keys. The underlying communication network is asynchronous and unreliable, allowing for message loss, delay, and reordering. These conditions reflect realistic UAV communication environments and are accounted for in the adversary model described in the next section.

3.2. Adversary Model

The adversary is modeled as a Dolev–Yao adversary with full control over the communication channel, as defined in [Dolev and Yao 1983]. It can intercept, modify, inject, and replay messages over both UAV–GCS and UAV–UAV links. The adversary is assumed to be computationally unbounded but constrained by the perfect cryptography assumption, i.e., it can only derive new messages from its current knowledge using the available cryptographic primitives. In addition, the adversary may compromise long-term credentials by revealing secret keys of honest parties at specific points in time. Honest parties, however, are assumed to execute the protocol faithfully. This threat model captures the open and distributed nature of UAV communication environments and motivates the need for strong security guarantees, including mutual authentication, key secrecy and forward secrecy.

3.3. Security Goals

Our primary objective is to establish secure communication channels between UAV–GCS and UAV–UAV links through an AKE protocol, serving as a foundational building block for subsequent secure data exchange. Under the adversary model described in Section 3.2, including possible long-term key compromise, we consider the following security goals:

Mutual Authentication. Both parties verify each other's identities before accepting a session. If an honest party accepts a session with a given peer identity and context, then

there is a corresponding execution of the peer with matching parameters. This prevents adversarially fabricated sessions and ensures agreement on the intended protocol run.

Session Key Secrecy. The established session key must remain unknown to an adversary that does not possess the honest parties' secret keys. In particular, the adversary should not be able to obtain the session key from observed, replayed, or modified protocol messages.

Forward Secrecy. Compromise of long-term secret keys after session completion must not reveal previously established session keys. Therefore, past session keys remain protected even if long-term credentials are exposed. This property is essential in UAV environments, where node capture or delayed key exposure may occur.

These properties provide a foundation for secure pairwise communication, supporting both UAV–GCS and UAV–UAV interaction patterns. We assume that cryptographic operations execute in an isolated environment; therefore, side-channel leakage and physical key extraction are outside the scope of the symbolic model.

4. Protocol Design

In this section, we introduce the EDHOC protocol and review the available post-quantum variants and their limitations, and specify the construction considered for this work.

4.1. EDHOC

Ephemeral Diffie–Hellman Over COSE (EDHOC) is a lightweight AKE protocol standardized by the Internet Engineering Task Force in RFC 9528 [Selander et al. 2024], designed specifically for constrained IoT environments. It is based on the MAC-then-Sign construction [Günther and Mukendi 2022] and minimizes communication overhead through the use of Concise Binary Object Representation (CBOR) encoding and CBOR Object Signing and Encryption (COSE) message structures. EDHOC follows a three-message design and achieves efficiency by tightly integrating key exchange, authentication, and transcript binding, making it well-suited for bandwidth-limited and resource-constrained devices [Fedrecheski et al. 2024].

Recent work has explored post-quantum extensions of EDHOC, notably through KEM-based constructions and hybrid designs [Fraile et al. 2026, Papon and Onete 2026], demonstrating that EDHOC's compact structure can be preserved under post-quantum assumptions. However, these proposals do not currently provide a formally verified model of their protocols. Regarding formal verification, [Nguyen 2025] introduces a signature-free KEM-only variant with symbolic verification in ProVerif, but does not address deployment in resource-constrained environments. Moreover, [Jacomme et al. 2023] comprehensively analyze the original EDHOC and explicitly identify the XOR encryption in Message 2 as malleable, requiring tailored equations to capture per-element modification attacks. This work addresses these gaps by replacing the MAC/XOR step with transcript-bound AEAD (eliminating this malleability and the auxiliary equations it requires) and by providing a formally verified model with a performance analysis targeting constrained UAV platforms.

4.2. Protocol Specification

The proposed protocol is inspired by post-quantum extensions of EDHOC, particularly the KEM-based constructions described in [Fraile et al. 2026, Papon and Onete 2026].

EDHOC originally adopts MAC authentication and XOR-based encryption to minimize message size in constrained IoT deployments. However, the XOR-based protection used in Message 2 has been identified as structurally malleable, being susceptible to per-element modification attacks [Jacomme et al. 2023]. In the post-quantum setting, this design choice becomes less attractive because the dominant communication cost is no longer the protected-field mechanism itself, but the size of public keys, ciphertexts, and signatures. For this reason, in contrast to existing post-quantum EDHOC designs, we replace the MAC/XOR construction used in Message 2 by transcript-bound AEAD. This removes the malleable XOR-based protection layer and enables confidentiality and integrity of protected fields to be represented through a single AEAD abstraction. The resulting composition is closer to modern secure-channel design, as exemplified by TLS 1.3 [Rescorla 2018], and has the benefit of yielding a more direct primitive composition.

A simplified message flow for the protocol is shown in Figure 2. Intermediate values are shown only at their initial computation and are recomputed by the other party when needed. In our use case, a UAV typically assumes the role of $\mathcal{I}$, while the GCS or a peer UAV acts as $\mathcal{R}$, serving as the contacted endpoint. Both parties possess long-term signature key pairs, denoted by $(sk_{\mathcal{I}}^{\mathsf{sig}}, pk_{\mathcal{I}}^{\mathsf{sig}})$ and $(sk_{\mathcal{R}}^{\mathsf{sig}}, pk_{\mathcal{R}}^{\mathsf{sig}})$, and are pre-provisioned with credentials issued by a trusted Certificate Authority (CA). Each credential binds an identifier to a long-term public signature key via a CA signature. Formally, the certificate of the initiator is defined as $\mathsf{Cert}_{\mathcal{I}} = (ID_{\mathcal{I}}, pk_{\mathcal{I}}^{\mathsf{sig}}, \sigma_{\mathsf{CA}})$, where $\sigma_{\mathsf{CA}} = \mathsf{Sign}(\langle ID_{\mathcal{I}}, pk_{\mathcal{I}}^{\mathsf{sig}} \rangle, sk_{\mathsf{CA}})$. An analogous definition holds for $\mathcal{R}$. In practice, certificates include additional metadata, such as validity periods, issuer information, and policy-related fields. These fields are omitted from the notation as they are not directly relevant to the cryptographic message flow or to the symbolic properties analyzed in this work. Both parties are assumed to possess an authentic copy of the CA public key pk_{CA}, which is used to verify certificates upon reception. The handshake follows a three-message structure:

Message 1. $\mathcal{I}$ generates a fresh KEM key pair and sends pk^{KEM} and the negotiated Cipher Suite (i.e., the set of algorithms used for encryption, signing, etc.) to $\mathcal{R}$. This first message establishes the initial protocol context and is included in the first transcript hash $TH_1 = \mathsf{H}(m_1)$.

Message 2. Upon receiving m_1, $\mathcal{R}$ encapsulates to the ephemeral KEM public key, obtaining the shared secret ss and corresponding ciphertext ct. The second transcript hash is then computed as $TH_2 = \mathsf{H}(TH_1||ct)$. To authenticate its identity and bind it to the current execution, $\mathcal{R}$ signs its certificate together with TH_2, obtaining σ_2, and constructs the authentication payload $P_2 = \langle \mathsf{Cert}_{\mathcal{R}}, \sigma_2 \rangle$. To protect the identity and authentication material, the payload P_2 is encrypted before transmission. For this purpose, $\mathcal{R}$ combines the shared secret ss with the transcript hash TH_2 to derive the intermediate pseudorandom key PRK_{2e}. It then derives the AEAD key K_2 and nonce IV_2 from PRK_{2e}, using distinct context labels for domain separation. Next, $\mathcal{R}$ encrypts P_2 using K_2 and IV_2, with TH_2 as associated data, obtaining the protected payload C_2. The second message is $m_2 = \langle ct, C_2 \rangle$.

Message 3. After receiving m_2, $\mathcal{I}$ decapsulates the KEM ciphertext using its ephemeral secret key, and recomputes PRK_{2e}, TH_2, and the AEAD decryption material. Next, $\mathcal{I}$ decrypts C_2, validates $\mathsf{Cert}_{\mathcal{R}}$ using the CA public key pk_{CA}, and verifies σ_2. If all these checks succeed, $\mathcal{I}$ authenticates $\mathcal{R}$ and prepares its own authentication payload. For this, it computes the transcript hash $TH_3 = \mathsf{H}(TH_2||P_2)$, signs its certificate together with

TH_3 using its long-term signing key, and obtains σ_3. It then constructs the authentication payload $P_3 = \langle \mathsf{Cert}_{\mathcal{I}}, \sigma_3 \rangle$. To encrypt P_3, $\mathcal{I}$ derives the AEAD key K_3 and nonce IV_3, using distinct context labels. Finally, encryption is done by using K_3 and IV_3, with TH_3 as associated data, obtaining the protected payload C_3. The third message is $m_3 = \langle C_3 \rangle$.

Upon receiving m_3, $\mathcal{R}$ derives the AEAD decryption material and decrypts C_3. It then validates $\mathsf{Cert}_{\mathcal{I}}$ and verifies σ_3. If the verification succeeds, both parties can derive the final shared key $PRK_{\mathsf{out}} = \mathsf{HKDF.Expand}(PRK_{2e}, TH_4, |PRK_{\mathsf{out}}|)$, where TH_4 represents the transcript hash obtained through $TH_4 = \mathsf{H}(TH_3||P_3)$. At this point, $\mathcal{I}$ and $\mathcal{R}$ have been mutually authenticated and share a fresh shared key bound to the complete protocol transcript.

5. Formal Verification

Formal verification has become an important tool for analyzing security protocols, since it allows security goals to be stated precisely and checked against well-defined adversary models [Barbosa et al. 2021]. Existing approaches include computational proof assistants, which provide security guarantees under explicit cryptographic assumptions but are harder to automate, and symbolic protocol verifiers, which abstract cryptographic primitives as ideal in order to reason automatically about protocol executions. Although symbolic models do not capture computational security, they provide rigorous guarantees against many protocol-level attacks. In this work, we adopt the symbolic approach and use TAMARIN for its expressiveness in modeling cryptographic primitives and security properties.

5.1. TAMARIN Prover

TAMARIN is a symbolic verification tool for the analysis of cryptographic protocols introduced by [Meier et al. 2013]. It models protocols as multiset rewriting systems and assumes a Dolev–Yao adversary, as described in Section 3.2. Security properties are specified as first-order logic lemmas over execution traces, enabling automated or interactive proofs. Its expressive framework and support for equational theories make it well suited for analyzing modern protocols, and it has been successfully employed for the formal verification of protocols such as TLS 1.3, 5G-AKA and IEEE 802.11 WPA2. TAMARIN supports both its native multiset rewriting formalism and higher-level protocol specifications through SAPIC$^+$ [Cheval et al. 2022], a stateful process language related to the applied π-calculus [Abadi et al. 2018]. In SAPIC$^+$, protocol roles are specified as concurrent processes, which are translated into TAMARIN's multiset rewriting rules via a semantics-preserving compilation. In this work, we use SAPIC$^+$ as a front-end modeling language and TAMARIN as the back-end verification engine. This allows for intuitive and structured modeling at the front-end, while leveraging the expressiveness and automated reasoning capabilities of TAMARIN at the back-end for rigorous security analysis.

5.2. Protocol Encoding

In SAPIC$^+$, protocol messages are represented as *terms*, abstracting their concrete bit-level representation and retaining only their symbolic structure. Terms are used to model basic values such as nonces, identifiers, long-term keys, and ephemeral secrets. Each role in the protocol (i.e., $\mathcal{I}$ and $\mathcal{R}$) is modeled as a *process* that performs a sequence of actions, including the generation of fresh values, message transmission and reception, and local computations. *Events* are used to record security-relevant actions during protocol

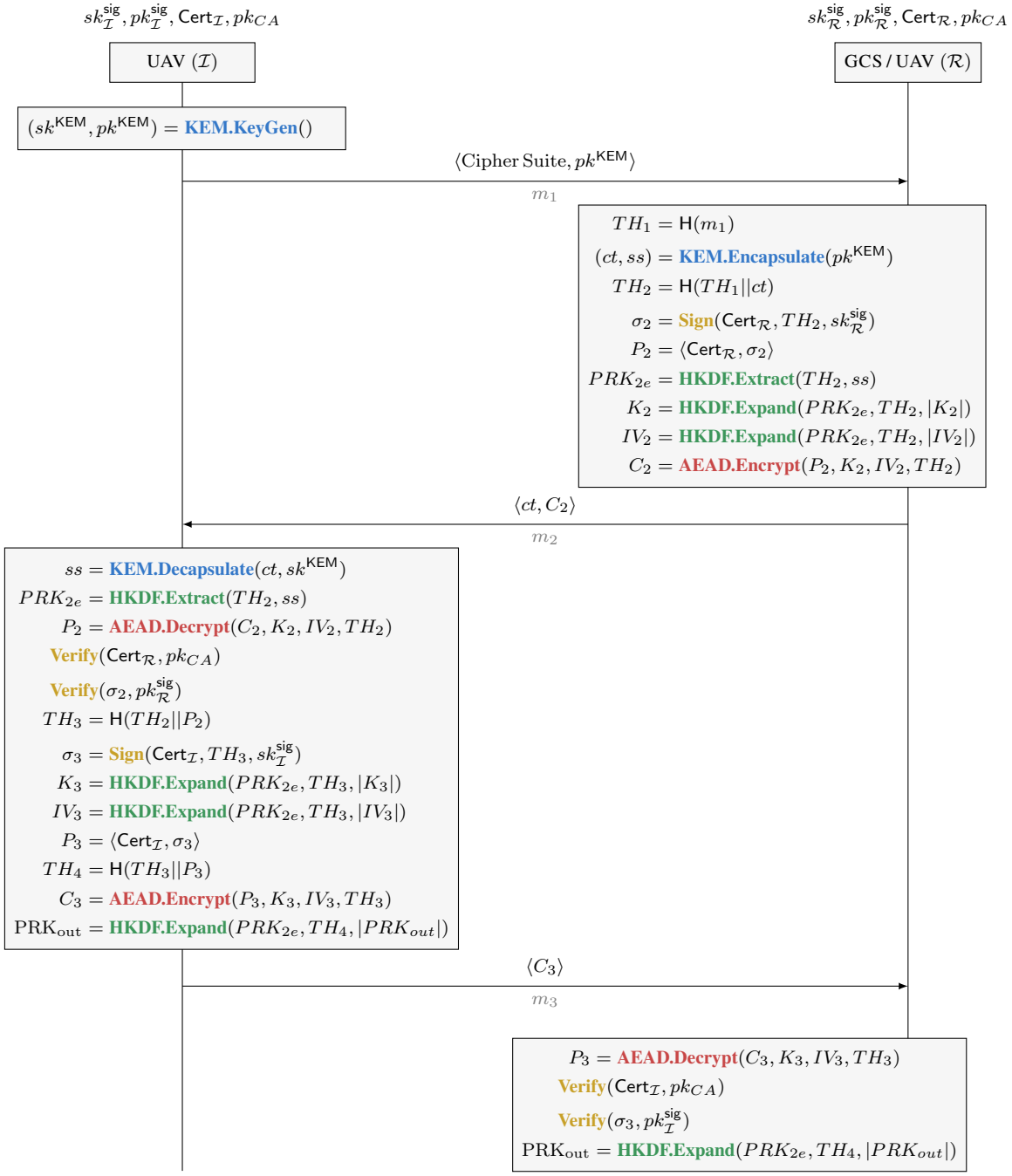

Figure 2. Simplified message flow of the proposed AKE protocol. Colors are used to represent the different cryptographic primitives. For clarity, labels, transcript hash recomputations, and auxiliary data fields are omitted.

execution and form the basis for specifying security properties. Each event is parameterized by session-specific values, such as identities and transcript data, allowing precise matching across protocol roles. The main events used in the protocol model are the following: **AcceptI** and **AcceptR** mark successful completion by $\mathcal{I}$ and $\mathcal{R}$, respectively, together with the derived session key, peer identities, and transcript values. **RunningI** and **RunningR** record ongoing protocol executions and serve as witnesses for authentication properties. **SkReveal**(k) denotes adversarial knowledge of a session key k and is used in secrecy

lemmas. Finally, **LtkRevealI**(pk) and **LtkRevealR**(pk) record compromise of the long-term secret key associated with pk, whereas **EphReveal**(pk_{KEM}) events record compromise of ephemeral KEM secret keys.

5.3. Cryptographic Primitives

In the symbolic model, all cryptographic primitives are represented as idealized operations. This approach treats cryptographic operations as perfect, allowing only their intended use while abstracting away computational details such as algorithmic structure and complexity. It also ensures that the symbolic analysis captures the essential security properties of the protocol while remaining agnostic to the specific implementation details of the underlying primitives. We now describe how each cryptographic primitive is modeled in TAMARIN.

Key Encapsulation Mechanism (KEM) is modeled using the function symbols **kempk**$/1$, **kemencap**$/2$ and **kemdecap**$/2$, where suffixes indicate the corresponding arity. The function **kempk**$(\mathtt{sk})$ derives the public encapsulation key from the secret key `sk`. Encapsulation is represented by **kemencap**$(\mathtt{ss}, \mathtt{pk})$, which produces a ciphertext embedding the shared secret `ss` under the public key `pk`. Decapsulation is modeled by the destructor **kemdecap**$(\mathtt{ct}, \mathtt{sk})$, which recovers the shared secret when applied to a valid ciphertext. Correctness of the KEM is captured by the equation **kemdecap**$($**kemencap**$(\mathtt{ss},$ **kempk**$(\mathtt{sk})), \mathtt{sk}) = \mathtt{ss}$. This ensures that only a party in possession of the corresponding secret key can recover the encapsulated secret. Decapsulation failures, re-encapsulation, and KEM binding properties remain outside the symbolic model and require computational analysis.

Digital Signatures are modeled using the function symbols **sign**$/3$, **pk**$/1$, and the destructor **sigverify**$/4$. A signature over a message `m` is represented as **sign**$(\mathtt{m}, \mathtt{r}, \mathtt{sk})$, where `sk` is the signing key and `r` denotes auxiliary randomness. The corresponding public verification key is obtained via **pk**$(\mathtt{sk})$. Signature verification is captured by the equation **sigverify**$($**sign**$(\mathtt{m}, \mathtt{r}, \mathtt{sk}), \mathtt{m}, \mathtt{r},$ **pk**$(\mathtt{sk})) = \mathsf{True}$. This enforces that only holders of the signing key can produce valid signatures, while any party with the corresponding public key can verify them. The model guarantees unforgeability under the symbolic assumption that the adversary cannot construct a valid signature without knowledge of the signing key.

Authenticated Encryption with Associated Data (AEAD) is modeled using the function symbols **aeadenc**$/4$ and **aeaddec**$/4$. Encryption is represented as **aeadenc**$(\mathtt{m}, \mathtt{k}, \mathtt{iv}, \mathtt{ad})$, where m is the plaintext, `k` is the symmetric key, `iv` is a nonce, and `ad` is the associated data. Decryption is modeled by the destructor **aeaddec**$(\mathtt{ct}, \mathtt{k}, \mathtt{iv}, \mathtt{ad})$. Correctness is expressed by the equation **aeaddec**$($**aeadenc**$(\mathtt{m}, \mathtt{k}, \mathtt{iv}, \mathtt{ad}), \mathtt{k}, \mathtt{iv}, \mathtt{ad}) = \mathtt{m}$. This abstraction captures both confidentiality and integrity: the adversary cannot recover the plaintext without the key, nor can it modify the ciphertext or associated data without causing decryption to fail.

Hash Functions and HMAC Key Derivation Functions (HKDF) are modeled using the function symbols **hash**$/1$, **hkdf_extract**$/2$, and **hkdf_expand**$/3$. In particular, **hash**$(\mathtt{m})$ represents a hash function, **hkdf_extract**$(\mathtt{salt}, \mathtt{ikm})$ models entropy extraction, **hkdf_expand**$(\mathtt{prk}, \mathtt{context}, \mathtt{length})$ derives keys bound to protocol transcripts. No equations are defined for these functions, reflecting the assumption that they behave as ideal cryptographic primitives. This ensures that derived keys are independent and indistinguishable unless the underlying inputs are known to the adversary. The use of transcript-dependent inputs in key derivation enforces binding between session keys and

the protocol execution, which is essential for achieving authentication and forward secrecy.

5.4. Security Properties

Security properties are specified as *lemmas*, which define logical constraints over event occurrences and must hold for all possible executions of the protocol, thereby capturing security guarantees in the presence of an active adversary. These lemmas are specified using first-order logic and verified using TAMARIN. Variables like $\#i$ denote trace timepoints, and relations such as $\#j < \#i$ denote temporal precedence. The variable `sid` denotes a local session identifier used to distinguish protocol executions in the model. If a lemma holds, TAMARIN proves that no execution trace violates it; otherwise, it produces a concrete counterexample trace demonstrating an attack. We next present the analyzed security properties, along with their corresponding lemma(s) and formalization in TAMARIN.

Mutual Authentication is formalized in terms of *non-injective agreement*, as defined in Lowe's hierarchy of authentication [Lowe 1997]. This property requires that, for any accepted session, there exists a corresponding execution of the peer that agrees on the relevant protocol parameters, such as identities and transcript values. This ensures that accepted sessions correspond to genuine protocol executions rather than to sessions fabricated by the adversary. Mutual authentication is captured by Lemmas 1 and 2.

Lemma 1 (Initiator → Responder Agreement)

If the initiator accepts a session, then a responder must have been running the protocol with the matching responder public key and intermediate transcript values.

$$\forall\, (\mathtt{sid_I}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{i}:$$
$$\mathbf{AcceptI}(\mathtt{sid_I}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{i}$$
$$\Rightarrow \exists\ \mathtt{sid_R}\ \#\mathtt{j}:$$
$$\mathbf{RunningR}(\mathtt{sid_R}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3})\ @\ \#\mathtt{j} \quad \& \quad \#\mathtt{j} < \#\mathtt{i}$$

Lemma 2 (Responder → Initiator Agreement)

If the responder accepts, then some initiator must have accepted the session with the matching initiator public key and transcript values.

$$\forall\, (\mathtt{sid_R}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{t}:$$
$$\mathbf{AcceptR}(\mathtt{sid_R}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{t}$$
$$\Rightarrow \exists\ \mathtt{sid_I}\ \#\mathtt{i}:$$
$$\mathbf{AcceptI}(\mathtt{sid_I}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{i} \quad \& \quad \#\mathtt{i} < \#\mathtt{t}$$

Protocol Correctness is captured by Lemma 3. This ensures that whenever both parties accept a session with matching transcript values (thus referring to the same protocol execution), they derive the same session key, guaranteeing consistency of the key establishment.

Lemma 3 (Key Agreement)

If both parties accept with the same transcript values, they derive the same session key.

$$\forall\, (\mathtt{sid_I}, \mathtt{sid_R}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k_I}, \mathtt{k_R})\ \#\mathtt{i}\ \#\mathtt{t}:$$
$$\mathbf{AcceptI}(\mathtt{sid_I}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k_I})\ @\ \#\mathtt{i} \quad \& \quad \mathbf{AcceptR}(\mathtt{sid_R}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k_R})\ @\ \#\mathtt{t}$$
$$\Rightarrow \mathtt{k_I} = \mathtt{k_R}$$

Session Key Secrecy is expressed by Lemmas 4 and 5. These properties state that if either the initiator or the responder reaches an acceptance state, then the corresponding session key cannot be derived by the adversary at any point in the execution.

Lemma 4 (Initiator Key Secrecy)

If the initiator accepts, then the derived session key remains unknown to the attacker.

$$\forall\ (\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{i}:$$
$$\mathbf{AcceptI}(\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{i}$$
$$\Rightarrow \neg\Big(\exists\ \#\mathtt{j}.\ \mathbf{SkReveal}(\mathtt{k})\ @\ \#\mathtt{j}\Big)$$

Lemma 5 (Responder Key Secrecy)

If the responder accepts, then the derived session key remains unknown to the attacker.

$$\forall\ (\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{t}:$$
$$\mathbf{AcceptR}(\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{t}$$
$$\Rightarrow \neg\Big(\exists\ \#\mathtt{j}.\ \mathbf{SkReveal}(\mathtt{k})\ @\ \#\mathtt{j}\Big)$$

Forward Secrecy is expressed by Lemmas 6 and 7. It requires that the session key remain unknown to an adversary even if long-term keys are compromised after the session completes [Cohn-Gordon et al. 2016]. We specify forward secrecy in relation to the long-term signature keys and ephemeral KEM secrets, and express it by stating that if an adversary learns an accepted session key, then either some long-term key was compromised *before* the session was accepted or the corresponding ephemeral KEM secret was revealed.

Lemma 6 (Initiator Forward Secrecy)

If the attacker learns an initiator-accepted session key, then either a long-term key was compromised before acceptance or the session's ephemeral KEM secret was revealed.

$$\forall\ (\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{pk_{KEM}}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{i}\ \#\mathtt{s}:$$
$$\mathbf{AcceptI}(\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{pk_{KEM}}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{i}\quad \&\quad \mathbf{SkReveal}(\mathtt{k})\ @\ \#\mathtt{s}$$
$$\Rightarrow \Big(\exists\ \#\mathtt{r}.\ \mathbf{LtkRevealI}(\mathtt{pk_I})\ @\ \#\mathtt{r}\ \ \&\ \ \#\mathtt{r} < \#\mathtt{i}\Big)\ |\ \Big(\exists\ \#\mathtt{r}.\ \mathbf{LtkRevealR}(\mathtt{pk_R})\ @\ \#\mathtt{r}\ \ \&\ \ \#\mathtt{r} < \#\mathtt{i}\Big)$$
$$|\ \Big(\exists\ \#\mathtt{r}.\ \mathbf{EphKEMReveal}(\mathtt{pk_{KEM}})\ @\ \#\mathtt{r}\ \ \&\ \ \#\mathtt{r} < \#\mathtt{i}\Big)$$

Lemma 7 (Responder Forward Secrecy)

If the attacker learns an responder-accepted session key, then either a long-term key was compromised before acceptance or the session's ephemeral KEM secret was revealed.

$$\forall\ (\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{pk_{KEM}}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{t}\ \#\mathtt{s}:$$
$$\mathbf{AcceptR}(\mathtt{sid}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{pk_{KEM}}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{t}\quad \&\quad \mathbf{SkReveal}(\mathtt{k})\ @\ \#\mathtt{s}$$
$$\Rightarrow \Big(\exists\ \#\mathtt{r}.\ \mathbf{LtkRevealI}(\mathtt{pk_I})\ @\ \#\mathtt{r}\ \ \&\ \ \#\mathtt{r} < \#\mathtt{t}\Big)\ |\ \Big(\exists\ \#\mathtt{r}.\ \mathbf{LtkRevealR}(\mathtt{pk_R})\ @\ \#\mathtt{r}\ \ \&\ \ \#\mathtt{r} < \#\mathtt{t}\Big)$$
$$|\ \Big(\exists\ \#\mathtt{r}.\ \mathbf{EphKEMReveal}(\mathtt{pk_{KEM}})\ @\ \#\mathtt{r}\ \ \&\ \ \#\mathtt{r} < \#\mathtt{t}\Big)$$

Executability is established in Lemma 8, which states that there exists at least one execution trace in which the protocol completes successfully between honest participants.

Lemma 8 (Protocol Executability)

There exists an honest execution in which both parties complete the protocol and derive the same session key without compromise.

$$\exists\ (\mathtt{sid_I}, \mathtt{sid_R}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ \#\mathtt{i}\ \#\mathtt{t}:$$
$$\mathbf{AcceptI}(\mathtt{sid_I}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{i}\quad \&\quad \mathbf{AcceptR}(\mathtt{sid_R}, \mathtt{pk_I}, \mathtt{pk_R}, \mathtt{th_2}, \mathtt{th_3}, \mathtt{th_4}, \mathtt{k})\ @\ \#\mathtt{t}$$
$$\&\ \neg\Big(\exists\ \#\mathtt{r}.\ \mathbf{LtkRevealI}(\mathtt{pk_I})\ @\ \#\mathtt{r}\Big)\quad \&\quad \neg\Big(\exists\ \#\mathtt{r}.\ \mathbf{LtkRevealR}(\mathtt{pk_R})\ @\ \#\mathtt{r}\Big)$$

This ensures that the proofs are not vacuous and that the specified roles, message flows, and constraints admit a valid execution. It serves as a consistency check and complements the security guarantees by confirming that they apply to realizable protocol runs.

5.5. Formal Verification Results

The verification was performed with TAMARIN version 1.13.0, using Maude version 3.5.1 as its rewriting back-end. The wellformedness checks were successful, meaning that TAMARIN did not report syntactic errors, malformed rules, or structural inconsistencies in the model. Experiments were executed on a virtual machine running Ubuntu 24.04.4 LTS, with four Intel Xeon Gold 6526Y CPU cores and 64 GiB of RAM. All lemmas were successfully verified, indicating that the proposed modifications to the protocol preserve the intended security properties. The complete verification took 15 min 45.35 s. In addition to the lemmas reported in Section 5.4, the verified model includes lemmas for freshness properties over transcript values and derived key material, supporting replay-resilience within the symbolic model, as well as explicit checks for Unknown Key Share (UKS) and Key-Compromise Impersonation (KCI) resistance. Due to space limitations, these additional lemmas are not presented in detail. A reproducible artifact containing the complete model and proof scripts is available on GitHub[1].

6. Performance Analysis

After establishing the symbolic security properties of the protocol, we estimate the cost of executing the handshake in two deployment scenarios: (i) a UAV–UAV exchange, where both endpoints are modeled as constrained ARM Cortex-M4 devices, and (ii) a UAV–GCS exchange, where the UAV is modeled as an ARM Cortex-M4 device and the GCS is modeled as an x86 Intel i7-4770 platform. The results are obtained by combining standardized primitive sizes, public benchmark cycle counts, and the protocol message structure described in Section 4. This provides only an initial cost estimate. A comprehensive comparison with other post-quantum EDHOC proposals remains to be conducted.

The evaluation considers ML-KEM and HQC-KEM as the KEM algorithms, and ML-DSA, FN-DSA and SLH-DSA for digital signatures. This selection uses the NIST Security Level 1 parameter sets available in each considered primitive family, while preserving diversity across lattice-based, code-based, and hash-based post-quantum constructions. The results indicate two aspects of the protocol – the communication cost indicates the number of bytes exchanged during the handshake, while the handshake latency reports the estimated cryptographic processing time for the UAV–UAV and UAV–GCS scenarios.

6.1. Communication Cost

Communication costs are derived from the protocol message format and from the standardized byte sizes of the underlying primitives. The message sizes are $|m_1| = |pk^{\mathsf{KEM}}| + \delta_1$, $|m_2| = |ct| + |pk^{\mathsf{sig}}| + |\sigma_2| + |\sigma_{\mathsf{CA}}| + |\mathsf{tag}_{\mathsf{AEAD}}| + \delta_2$ and $|m_3| = |pk^{\mathsf{sig}}| + |\sigma_3| + |\sigma_{\mathsf{CA}}| + |\mathsf{tag}_{\mathsf{AEAD}}| + \delta_3$, where Ascon-AEAD128 uses a 16-byte authentication tag, and each δ_i representes the non-cryptographic overhead from CBOR metadata, identities, and cipher suite negotiation data. We omit these terms from the main estimates to isolate the dominant contribution of the cryptographic material The classical baseline follows the same abstract

[1]https://github.com/regras/pq-edhoc-tamarin

message structure, replacing the KEM public key and ciphertext with ECDH P-256 values and the post-quantum signature keys and signatures with ECDSA P-256 values.

To estimate the communication delay, we compute the serialization delay required to transmit the protocol payloads under two representative link rates: 250 kbps and 1 Mbps. For a message size of B bytes and transmission rate R bits/s, the serialization delay is given by $T_{\text{tx}} = 8B/R$. This estimate does not include propagation delay, MAC-layer contention, fragmentation, retransmissions, queueing, or processing time and should therefore be interpreted as a lower bound on transmission latency, rather than as an estimate of end-to-end latency in an operational deployment. We present the estimated size of each message and communication cost for each cipher suite in Table 1.

Table 1. Estimated communication cost and serialization delay for different cipher suites. ECDH/ECDSA P-256 values are given as a traditional baseline. Message sizes are given in bytes, and delays account only for payload transmission time.

Cipher Suite	$\lvert m_1 \rvert$	$\lvert m_2 \rvert$	$\lvert m_3 \rvert$	Total	T_{tx} @ 250 kbps	T_{tx} @ 1 Mbps
ECDH P-256 + ECDSA P-256	64	272	208	544	17.41 ms	4.35 ms
ML-KEM-512 + FN-DSA-512	800	3,013	2,245	6,058	193.86 ms	48.46 ms
ML-KEM-512 + ML-DSA-44	800	6,936	6,168	13,904	444.93 ms	111.23 ms
ML-KEM-512 + SLH-DSA-128f	800	34,992	34,224	70,016	2,240.51 ms	560.13 ms
HQC-KEM-128 + FN-DSA-512	2,249	6,678	2,245	11,172	357.50 ms	89.38 ms
HQC-KEM-128 + ML-DSA-44	2,249	10,601	6,168	19,018	608.58 ms	152.14 ms
HQC-KEM-128 + SLH-DSA-128f	2,249	38,657	34,224	75,130	2,404.16 ms	601.04 ms

The message m_1 depends only on the selected KEM public key and is therefore smaller for ML-KEM-512 than for HQC-KEM-128. In contrast, the signature scheme primarily affects m_2 and m_3, since both carry authenticated payloads containing public keys and signatures. Therefore, FN-DSA-512 yields the smallest handshake sizes for post-quantum primitives because of its compact public keys and signatures, whereas SLH-DSA-128f produces the largest overhead due to its larger signatures. However, $|m_3|$ remains unchanged for a fixed signature scheme, since m_3 does not carry KEM material.

6.2. Handshake Latency

We estimate the handshake latency of the protocol by combining the cryptographic operation counts with benchmark results for the selected post-quantum primitives. The objective of this analysis is not to provide an end-to-end latency measurement, but rather to isolate the computational cost introduced by the post-quantum cryptographic operations. The protocol roles are asymmetric with respect to the KEM operations. The initiator generates a fresh KEM key pair and later decapsulates the responder's ciphertext, while the responder encapsulates to the initiator's KEM public key. Both parties generate one digital signature and perform one verification for the peer's protocol signature and one verification for the peer's certificate. Therefore, the public-key cost for each role is modeled as $T_{\mathcal{I}} = T_{\mathsf{KEM.KeyGen}} + T_{\mathsf{KEM.Decaps}} + T_{\mathsf{Sign}} + 2\,T_{\mathsf{Verify}}$ and $T_{\mathcal{R}} = T_{\mathsf{KEM.Encaps}} + T_{\mathsf{Sign}} + 2\,T_{\mathsf{Verify}}$. The total cryptographic latency of one complete handshake is then given by $T_{\mathsf{HS}} = T_{\mathcal{I}} + T_{\mathcal{R}}$.

The benchmark data comes from public cycle-count measurements. For the UAV endpoints, we use the pqm4 benchmark suite [Kannwischer et al. 2024], which targets ARM Cortex-M4 microcontrollers and reports cycle counts for post-quantum

KEM and signature operations. For the GCS endpoint, we use the SUPERCOP/eBATS [Bernstein and Lange 2024] measurements for an amd64 Haswell platform equipped with an Intel Core i7-4770. For the traditional baseline, we use the optimized ARM Cortex-M4 benchmark implementation of ECDH/ECDSA P-256 by [Lenngren 2021]. The cycle counts are then converted to milliseconds using assumed processor frequencies. Table 2 reports the estimated cryptographic processing latency for both deployment scenarios.

Table 2. Estimated cryptographic processing latency for UAV–UAV and UAV–GCS communication. The UAV is modeled as an ARM Cortex-M4 at 168 MHz, and the GCS is modeled as an Intel i7-4770 at 3.4 GHz. Times are given in milliseconds.

Cipher Suite	UAV ($\mathcal{I}$)	UAV ($\mathcal{R}$)	GCS ($\mathcal{R}$)	UAV–UAV	UAV–GCS
ECDH P-256 + ECDSA P-256	19.24	19.24	0.35	38.49	19.59
ML-KEM-512 + FN-DSA-512	312.32	307.66	0.17	619.98	312.49
ML-KEM-512 + ML-DSA-44	80.57	75.91	0.17	156.48	80.74
ML-KEM-512 + SLH-DSA-128f	2,463.73	2,459.07	18.58	4,922.80	2,482.31
HQC-KEM-128 + FN-DSA-512	1,567.02	932.36	0.23	2,499.39	1,567.26
HQC-KEM-128 + ML-DSA-44	1,335.28	700.61	0.23	2,035.89	1,335.51
HQC-KEM-128 + SLH-DSA-128f	3,718.43	3,083.77	18.65	6,802.20	3,737.08

The results show how primitive selection affects handshake feasibility. Among the evaluated combinations, ML-KEM-512 + ML-DSA-44 provides the most balanced performance profile, requiring approximately 156 ms for a UAV–UAV handshake and 81 ms in the UAV–GCS case. This reduction occurs because responder operations become almost negligible on the i7 platform and the total cost is dominated by the constrained UAV initiator. HQC-KEM-128 and SLH-DSA-128f provide useful comparison baselines, but their estimated costs make them unsuitable as default choices for low-latency scenarios. FN-DSA-512 remains interesting for bandwidth-constrained deployments, but its suitability depends on the availability of optimized implementations for the target microcontroller. It is worth noting that the proposed protocol is not intended for per-packet protection. Once the parties authenticate and derive a shared session key, subsequent traffic can be protected using lightweight symmetric cryptography. Therefore, a sub-second cost may be acceptable for initialization or occasional rekeying before deriving longer-lived pairwise or group keys. However, the results also show that the protocol should not be placed on a hard real-time control path or executed repeatedly for every data exchange.

7. Conclusion

This work presents a post-quantum variant of the EDHOC protocol for pairwise UAV–GCS and UAV–UAV links. Its security properties were analyzed through a symbolic model in TAMARIN and verified under a Dolev–Yao adversary. The results show that, for the ML-KEM-512 + ML-DSA-44 instantiation, the protocol introduces non-negligible but manageable communication and computational costs. These estimates suggest that the costs remain feasible in UAV–GCS and UAV–UAV links. The security properties established in this work hold within the symbolic model. While symbolic verification does not directly capture computational security, it provides rigorous guarantees against a broad class of protocol-level attacks. Moreover, we are extending the analysis to identity-protection properties through observational equivalence, with the results to be reported in a future publication. Future work will expand the quantitative comparison and benchmark the protocol on representative UAV hardware under realistic operating conditions.

References

Abadi, M., Blanchet, B., and Fournet, C. (2018). The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM*, 65(1):1–41.

Aissaoui, R. (2024). *Assessment and Optimization of Post-Quantum Cryptographic Protocols for Civil UAV Communications*. PhD thesis, École Nationale de l'Aviation Civile. Theses.fr ID: 2024ENAC0009.

Barbosa, M., Barthe, G., Bhargavan, K., Blanchet, B., Cremers, C., Liao, K., and Parno, B. (2021). SoK: Computer-aided cryptography. *IEEE Symposium on Security and Privacy*.

Bernstein, D. J. and Lange, T. (2024). eBACS: ECRYPT benchmarking of cryptographic systems. https://bench.cr.yp.to. Accessed: 2026-05-07.

Blanchet, B. (2016). Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1–2):1–135.

Boneh, D. and Shoup, V. (2023). A graduate course in applied cryptography. https://toc.cryptobook.us/. Version 0.6.

Boyd, C. and Mathuria, A. (2019). *Protocols for Authentication and Key Establishment*. Springer, 2 edition.

Cheval, V., Jacomme, C., and Kremer, S. (2022). SAPIC+: Protocol verifiers of the world, unite! In *31st USENIX Security Symposium (USENIX Security 2022)*.

Cohn-Gordon, K., Cremers, C., and Garratt, L. (2016). Post-compromise security. Cryptology ePrint Archive, Paper 2016/221.

Dolev, D. and Yao, A. C. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207.

Fedrecheski, G., Vučinić, M., and Watteyne, T. (2024). Performance comparison of EDHOC and DTLS 1.3 in internet-of-things environments. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6.

Fraile, L. P., Koulamas, C., Fournaris, A. P., and Haleplidis, E. (2026). KEM-based authentication for EDHOC in initiator-known responder (IKR) scenarios. Internet-Draft draft-pocero-authkem-ikr-edhoc-02, Internet Engineering Task Force. Work in Progress.

Günther, F. and Mukendi, M. I. T. (2022). Careful with MAC-then-Sign: A computational analysis of the EDHOC lightweight authenticated key exchange protocol. Cryptology ePrint Archive, Paper 2022/1705.

He, L., Zhao, M., Wang, X., Wang, J., Wang, Z., and Liu, S. (2025). A post-quantum authentication and key agreement scheme for drone swarms. *Electronics*, 14(17).

Jacomme, C., Cremers, C., Bhargavan, K., Cherrier, S., Fouque, P.-A., and Lahiani, S. (2023). A comprehensive formal security analysis of EDHOC. In *32nd USENIX Security Symposium (USENIX Security 2023)*, pages 2607–2624, Anaheim, CA. USENIX Association.

Kannwischer, M. J., Petri, R., Rijneveld, J., Schwabe, P., and Stoffelen, K. (2024). PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4.

Lenngren, E. (2021). P256-Cortex-M4: P-256 ECDSA / ECDH for Cortex-M4 and Cortex-M33. https://github.com/Emill/P256-Cortex-M4. GitHub repository. Accessed: 2026-05-20.

Lowe, G. (1997). A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Workshop on Computer Security Foundations*, CSFW '97, page 31, USA. IEEE Computer Society.

Meier, S., Schmidt, B., Cremers, C., and Basin, D. (2013). The Tamarin prover for the symbolic analysis of security protocols. In *Computer Aided Verification – CAV 2013*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer.

Nguyen, C. (2025). A formally verified quantum-safe key exchange protocol. Master's thesis, Aalto University. Aaltodoc URN: urn:nbn:fi:aalto-202512179381.

NIST (2024a). Module-lattice-based digital signature standard (ML-DSA). Federal Information Processing Standards Publication FIPS 204, NIST.

NIST (2024b). Module-lattice-based key-encapsulation mechanism standard (ML-KEM). Federal Information Processing Standards Publication FIPS 203, NIST.

NIST (2024c). Stateless hash-based digital signature standard. Federal Information Processing Standards Publication FIPS 205, NIST.

NIST (2025a). NIST selects HQC as fifth algorithm for post-quantum encryption. https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption. Accessed: 2026-05-11.

NIST (2025b). Post-quantum cryptography standardization. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization. Accessed: 2026-05-11.

Papon, C. and Onete, C. (2026). Post-quantum EDHOC: Initiator and responder using signature and/or KEM. Internet-Draft draft-papon-lake-pq-edhoc-00, Internet Engineering Task Force. Work in Progress.

Rescorla, E. (2018). The transport layer security (TLS) protocol version 1.3. RFC 8446, RFC Editor.

Selander, G., Mattsson, J., Palombini, F., and Seitz, L. (2024). Ephemeral Diffie-Hellman over COSE (EDHOC). RFC 9528, Internet Engineering Task Force.

Wu, Y., Jia, Z., Wu, Q., and Zhu, Y. (2025). A lightweight authentication and key agreement protocol design for FANET.

Xia, T., Wang, M., He, J., Yang, G., Fan, L., and Wei, G. (2024). A quantum-resistant identity authentication and key agreement scheme for UAV networks based on Kyber algorithm. *Drones*, 8(8).