applied sciences

MDPI

Article

A Lightweight Hybrid Post-Quantum Authentication Protocol for Resource-Constrained UAV–GCS Communications

Lin Lin [1,2,*], Jiaguo Tang [1], Jiahao Liao [1], Ruiqi Zhang [1] and Haibin Wang [2,3]

[1] The Key Laboratory of Flight Techniques and Flight Safety, Civil Aviation Flight University of China, Guanghan 618307, China; tangjiaguo@cafuc.edu.cn (J.T.); liaojiahao@cafuc.edu.cn (J.L.); rachel@cafuc.edu.cn (R.Z.)
[2] College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; wanghaibin123@cafuc.edu.cn
[3] Civil Aviation Safety Engineering College, Civil Aviation Flight University of China, Guanghan 618307, China
* Correspondence: linlin@cafuc.edu.cn

Abstract

Resource-constrained unmanned aerial vehicle–ground control station (UAV–GCS) communications face substantial post-quantum authentication costs, expensive cryptographic operations triggered by invalid requests, and key-state desynchronization during rekeying. To address these problems, this paper proposes QUAD, a lightweight hybrid post-quantum authentication and key agreement protocol. QUAD combines request pre-authentication, hybrid key establishment, and mutual key confirmation to reject invalid requests before costly cryptographic operations. It also allows a bounded transition period for receiving records under the old and new keys to mitigate key-state inconsistencies caused by link disruptions. Confidentiality and authentication are analyzed using ProVerif, while cryptographic and communication overhead, denial-of-service scenarios, and key-update behavior are evaluated on a Raspberry Pi platform. Under the specified security model, QUAD satisfies the tested authentication and confidentiality properties, reduces the computational burden of invalid requests, and improves bidirectional communication continuity under selected packet-loss conditions. These findings support its feasibility for post-quantum security in resource-constrained UAV–GCS links.

Keywords: unmanned aerial vehicles; post-quantum cryptography; authentication; key agreement; pre-authentication; key update

1. Introduction

Unmanned aerial vehicles (UAVs) are increasingly used for inspection, logistics, and emergency communications. Consequently, the command-and-control (C2) link between a UAV and its ground control station (GCS) is essential to reliable flight operations [1]. Because wireless channels are open, these links are exposed to eavesdropping, replay, message tampering, impersonation, and denial-of-service (DoS) attacks [2]. Advances in large-scale quantum computing also pose a potential threat to conventional public-key cryptography: algorithms such as RSA and elliptic-curve cryptography (ECC), whose security relies on integer factorization or discrete logarithms, would be vulnerable to sufficiently capable quantum computers [3]. Secure communications for resource-constrained UAVs must therefore address both current network attacks and future post-quantum key-establishment requirements.

Check for updates

Academic Editor: Juan-Carlos Cano

Received: 7 September 2026
Revised: 15 October 2026
Accepted: 26 October 2026
Published: 28 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Communication reliability is closely related to flight control and cooperative missions. Nowakowski et al. [4] investigated UAV-assisted reconnaissance and path planning for unmanned ground vehicles (UGVs); Espinosa Pedroza et al. [5] studied robust leader–follower consensus control under external disturbances; and Borja-Jaimes et al. [6] improved quadrotor trajectory tracking through disturbance observation and compensation. These studies demonstrate the importance of reliable information exchange for cooperative missions and flight control, although they do not directly investigate communication authentication protocols. In UAV–GCS C2 links, packet loss, varying delays, and key-state desynchronization during rekeying may disrupt the reception of telemetry and control commands. Communication continuity under abnormal link conditions must therefore be considered.

To reduce authentication overhead on resource-constrained UAVs, Li et al. [7] combined physical unclonable functions (PUFs) with ASCON to design an authentication protocol supporting UAV–GCS and UAV–UAV communications. Meanwhile, post-quantum key establishment must also account for the additional costs of cryptographic operations and handshake communication. Regarding key management, Wang et al. [8] investigated group key agreement for UAV swarms, while Xiao et al. [9] formally analyzed key update and authentication properties of ratchet protocols. However, these studies do not directly address key-state desynchronization caused by lost acknowledgment messages, delayed packets, or repeated requests in point-to-point UAV–GCS links. Therefore, protocol design for resource-constrained UAV–GCS communications should consider not only post-quantum key establishment and authentication overhead, but also explicit state transitions and exception-handling rules during rekeying.

To address these challenges, this paper proposes QUAD, a lightweight hybrid post-quantum authentication and key agreement protocol for resource-constrained UAV–GCS C2 links. Before ML-KEM decapsulation, QUAD performs Cookie verification, HMAC verification based on a long-term shared key (LTSK), and handshake-transaction deduplication. Finished messages provide mutual key confirmation, while a protected SessionReady record activates the session. During rekeying, staged transmission-key switching and a bounded period of overlapping reception of old and new epochs mitigate desynchronization caused by link impairments. ProVerif-based formal verification and experiments on a Raspberry Pi platform evaluate the protocol's security properties, computational and communication overhead, abnormal request handling, and bidirectional communication continuity.

The main contributions are as follows:

1. A hybrid post-quantum authentication protocol is designed for resource-constrained UAV–GCS links. ML-KEM-768 and a preconfigured LTSK establish an authenticated session; transcript binding, direction-separated keys, and Finished verification protect bidirectional communication. Computational overhead, handshake latency, and communication overhead are measured experimentally.
2. A layered pre-decapsulation request-checking mechanism is designed, with Cookie verification, LTSK-HMAC verification, and handshake-transaction deduplication performed in sequence. Initial-request flooding, invalid Cookies, valid Cookies with incorrect MACs, and duplicate M3 messages are tested to evaluate filtering effectiveness, resource consumption, and legitimate-handshake performance. The focus is on the processing order and practical costs of established authentication mechanisms.
3. An R1–R4 four-message rekeying procedure is developed with explicit rules for retransmissions, transaction deduplication, key switching, timeouts, and old-key deletion. Bounded reception of both old- and new-epoch packets addresses temporary key-state desynchronization. Packet-loss, delay, ablation, and Single-key baseline experiments evaluate telemetry and control-command reception.

4. Based on the Dolev–Yao adversary model [10], ProVerif [11] is used to examine handshake confidentiality, mutual authentication, key confirmation, rekeying event correspondences, and selected long-term secret compromise scenarios. Implementation experiments are further used to evaluate protocol performance and its applicable conditions.

2. Related Work

Existing research on secure UAV communication primarily addresses lightweight authentication, post-quantum key agreement, and key updates during sustained communication. This section reviews these three areas and discusses their applicability to resource-constrained UAV–GCS links.

2.1. Lightweight Authentication

Under constraints on computation, memory, and energy, lightweight UAV authentication commonly employs ECC, hash functions, HMAC, dynamic identities, and physical unclonable functions (PUFs). Wazid et al. [12] proposed a remote-user authentication and key agreement scheme for the Internet of Drones (IoD); Hussain et al. [13] designed an ECC-based UAV authentication scheme; Srinivas et al. [14] employed temporary credentials and anonymous identities; Jan et al. [15] investigated an IoD authentication scheme based on HMAC and hashing; and Park et al. [16] used PUFs to protect UAV identities.

Related studies also address access control and cooperative UAV communications. Bera et al. [17] examined access control in UAV-assisted IoT environments, while Shariq et al. [18] and Illyass et al. [19] proposed authentication mechanisms for inter-UAV and swarm networks, respectively. MAVLink 2 message signing authenticates message origins with shared keys and timestamps and mitigates replay attacks [20], but does not provide post-quantum key establishment or data encryption.

These schemes reduce authentication overhead, but protocols relying on ECC for key establishment remain vulnerable to future quantum attacks, while hash- or HMAC-based authentication cannot replace post-quantum key establishment. Furthermore, relatively few studies jointly address abnormal request filtering before post-quantum decapsulation and the impact of rekeying desynchronization on bidirectional data reception.

2.2. Post-Quantum Authentication and Key Agreement

Post-quantum authentication and key agreement mainly rely on lattice cryptography, post-quantum signatures, and key encapsulation mechanisms (KEMs). Bos et al. [21] proposed CRYSTALS-Kyber, whose standardized successor, ML-KEM, is specified in NIST FIPS 203 and includes the ML-KEM-768 parameter set used here [22]. Tawfeeq et al. [23] investigated a Dilithium signature-based IoD authentication scheme, while He et al. [24] applied Kyber to UAV group key agreement and authentication.

For different IoD applications, Nair et al. [25] combined PUFs and post-quantum cryptography for cross-domain authentication; Xu et al. [26] studied quantum-resistant certificateless group authentication for large-scale IoT devices; and Mishra et al. [27] proposed a lattice-based secure UAV communication scheme. Xia et al. [28] combined Kyber, HMAC, and ECDH for UAV authentication and key agreement, while Chaudhary and Lee [29] used ring learning with errors (RLWE) to construct an IoD authentication and key agreement protocol supporting dynamic identities. These schemes differ in trust configuration, number of participants, and computational overhead.

For general-purpose secure communication, KEMTLS replaces conventional handshake signatures with a KEM for server authentication [30]. Han et al. [31] studied security constructions for KEM-based authenticated key exchange; Alwen et al. [32] formally an-

alyzed HPKE authentication modes and key derivation; Zhou et al. [33] discussed the conditions under which CPA-secure KEMs can be used in post-quantum TLS 1.3; and Liu et al. [34] proposed the CuKEM hybrid encapsulation framework and analyzed its security notions and efficiency.

Existing post-quantum schemes may increase online signing and verification costs, and some target UAV swarms, cross-domain authentication, or general TLS environments rather than resource-constrained point-to-point UAV–GCS links. QUAD does not present ML-KEM, HMAC, or Finished as new cryptographic mechanisms; instead, it focuses on layered checks before decapsulation and key-state switching during bidirectional communication.

2.3. Key Updates and Communication-State Maintenance

Long-running UAV–GCS communication requires continuous key-state management. Xu et al. [35] proposed a continuous group key agreement scheme for vehicular networks, employing key evolution to address long-term security; Ko et al. [36] investigated a UAV communication protocol incorporating key management. These studies primarily focus on key updates or group management, whereas packet loss, delay, and missing acknowledgments may temporarily leave the communicating parties in different key states.

Established secure communication protocols already incorporate Cookies and key update mechanisms. IKEv2 uses Cookies to limit the resource costs of forged initial requests and defines security-association updates and retransmission handling [37]. DTLS 1.3 provides retransmission, acknowledgment, replay detection, and key update mechanisms as well as Cookie protection [38]. WireGuard uses Cookies, timers, and session-key management to maintain secure communication [39]. Rosenpass incorporates a Cookie mechanism to reduce state-disruption risks and can update post-quantum shared keys in combination with WireGuard [40]. EDHOC employs compact encoding and authenticated key exchange and specifies message deduplication and related processing rules [41].

Existing studies improve communication security through lightweight authentication, post-quantum key establishment, and continuous key management. However, resource-constrained UAV–GCS links still require efficient abnormal-request handling and bidirectional communication during rekeying. QUAD addresses these issues using layered pre-decapsulation checks, a confirmed rekeying procedure, and bounded old/new-epoch reception, whose effects are evaluated by subsequent security analyses and experiments.

3. UAV Communication Model and Security Requirements

UAV C2 links face forged control commands, replay and man-in-the-middle attacks, telemetry tampering, and denial-of-service threats. This section defines QUAD's communication model, adversarial capabilities, and security requirements as a basis for the protocol design and formal verification.

3.1. System Communication Model

Figure 1 depicts a C2 link between one UAV and a ground control station (GCS), the two entities considered in this work.

1. Ground control station: As a trusted control node, the GCS manages system parameters and keys, issues control commands, and stores the ML-KEM private key and per-UAV long-term shared keys. It also performs identity verification, shared-secret decapsulation, and session-key derivation.
2. Unmanned aerial vehicle: As a resource-constrained mobile node, the UAV exchanges telemetry and control commands with the GCS over an open wireless channel. Communication comprises offline provisioning and online operation.

- Offline provisioning: Before a mission, trusted interfaces such as USB/UART are used to provision each UAV with a dedicated long-term shared key (LTSK), pseudonymous identity (PID), GCS ML-KEM public key, and GCS identity information. The GCS stores the corresponding LTSK and its ML-KEM private key.
- Online communication: The UAV and GCS authenticate each other and negotiate keys over an open wireless channel, protect telemetry and control commands with separate direction-specific encryption keys, and perform key updates as required.

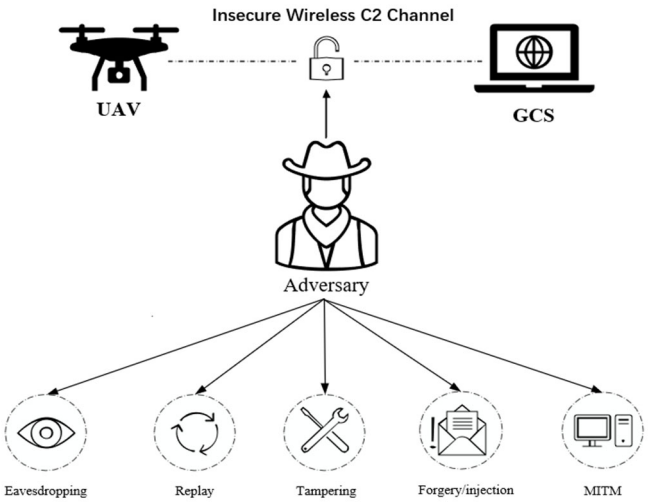


Figure 1. UAV–GCS communication and threat model.

3.2. Threat Model

QUAD is analyzed using the Dolev–Yao (DY) model. The adversary A completely controls the public channel between the UAV and GCS and can construct messages from known information, but cannot violate the model's ideal cryptographic rules. Quantum computing is considered a design motivation; ProVerif does not directly simulate Shor's algorithm.

The adversary A has the following capabilities:

1. Network control: The adversary can eavesdrop on, intercept, modify, delete, inject, and replay messages on the public channel.
2. Forgery and impersonation: The adversary can construct forged handshake or data records and attempt to impersonate a legitimate UAV or GCS.
3. Quantum-computing threat: A sufficiently powerful quantum computer could use Shor's algorithm to attack RSA, ECC, and Diffie–Hellman. QUAD uses ML-KEM-768 to establish a shared secret, but formal verification does not directly prove the quantum resistance of the underlying cryptographic algorithm.

The adversary A is subject to the following cryptographic assumptions:

- The adversary cannot efficiently solve the module learning with errors (MLWE) problem and therefore cannot recover the ML-KEM shared secret from its ciphertext without the corresponding private key.
- AES-256, SHA-256, HMAC-SHA-256, and HKDF-SHA-256 are assumed to satisfy their respective security requirements. ProVerif abstracts them as ideal cryptographic primitives without directly proving their computational security.
- In the default model, the adversary cannot obtain the LTSK, the GCS ML-KEM private key, or the Cookie master key. Section 5 separately considers compromise scenarios, including historical-session confidentiality after a single secret is disclosed follow-

ing M6, as well as predefined dual disclosure and future impersonation after LTSK compromise. These results do not establish full forward secrecy.

3.3. Security Requirements

The proposed protocol is designed to meet the following requirements for open, resource-constrained C2 links:

1. Mutual authentication: The UAV and GCS must authenticate each other before a session is activated to prevent impersonation. QUAD authenticates the UAV using LTSK-HMAC and achieves mutual key confirmation using Finished messages derived from the current handshake keys.
2. Session-key confidentiality and post-quantum security: An adversary should not be able to derive session keys by intercepting, modifying, or replaying handshake messages. QUAD combines an ML-KEM shared secret with LTSK-derived material to establish session keys under the stated cryptographic assumptions.
3. Data confidentiality and integrity: Telemetry and control commands must be encrypted and integrity protected. QUAD uses separate AES-256-GCM keys for the two directions and authenticates the record header as additional authenticated data to prevent unauthorized modification.
4. Replay and man-in-the-middle resistance: The protocol must prevent replay of historical records and forgery or tampering of handshake messages. QUAD protects authentication with nonces, Cookies, MACs, and handshake-context binding, while SID, Epoch, Seq, and independent replay windows reject duplicate data records.
5. Denial-of-service resistance: Invalid or duplicate requests should not repeatedly trigger expensive operations. QUAD performs Cookie verification, LTSK-HMAC verification, and transaction deduplication before ML-KEM decapsulation to reduce processing costs for abnormal requests, although this cannot eliminate all DoS attacks.
6. Rekeying synchronization: The protocol should reduce key-state desynchronization caused by packet loss, delays, and missing acknowledgments. QUAD defines switching conditions through R1–R4 and permits state-valid old- and new-epoch records during a bounded transition, with separate replay windows, to reduce bidirectional data losses. However, uninterrupted communication cannot be guaranteed when acknowledgments are persistently lost.

4. Protocol Design

QUAD authenticates identities using a long-term shared key, establishes a shared secret with ML-KEM, and derives distinct encryption keys for telemetry and control commands. This section specifies initial handshake, data protection, and rekeying rules.

4.1. Notation and Parameters

The protocol uses ML-KEM-768, SHA-256, HMAC-SHA-256, HKDF-SHA-256, and AES-256-GCM. $E(\cdot)$ denotes an unambiguous encoding with defined field order, lengths, and byte order; I2OSP(x,n) encodes an integer x as an n-byte big-endian value. L denotes a fixed public, purpose-specific domain-separation label whose value is defined by the protocol and is not transmitted as a separate field. Table 1 summarizes the notation.

Table 1. Main QUAD symbols and parameters.

Symbol	Definition and Purpose
PID_i, $LTSK_i$	UAV_i pseudonym and unique long-term key
MK, Cookie	GCS Cookie master key and token
pk_G, sk_G	GCS ML-KEM public and private keys
ct, ss; ct′, ss′	Initial and rekey ciphertexts and shared secrets
MAC_U, HID	UAV authentication MAC and handshake ID
TH_3, TH_4, TH_5	Handshake transcript hashes
K_{sym}, PRK	LTSK-derived material and hybrid root key
$K_{Fin,G/U}$, $Finished_{G/U}$	Peer confirmation keys and values
SID, e, e', Seq	Session ID, epochs, and sequence number
ES_e, K^e_{U2G}, K^e_{G2U}	Epoch secret and directional traffic keys
RID, RH	Rekey transaction ID and transcript hash
RPRK′, K_{RK}, KC_G	Rekey root key, confirmation key, and GCS tag
$E(\cdot)$, L_x	Canonical encoding and domain-separation labels

4.2. Protocol Overview

As specified in the offline provisioning procedure in Section 3.1, the UAV holds PID_i, its dedicated $LTSK_i$, the GCS public key pk_G, and GCS identity ID_G. The GCS stores the matching $LTSK_i$ and private key sk_G and generates a Cookie master key MK. UAVs never share an $LTSK_i$, and private keys and shared secrets are not transmitted over the wireless channel. The GCS obtains the source IP address locally; ID_G is not transmitted in M3.

Figure 2 summarizes QUAD: M1–M2 obtain a Cookie, M3–M5 provide authentication and mutual key confirmation, and M6 activates the session. The parties subsequently transmit data under separate direction-specific keys and perform key updates using R1–R4.

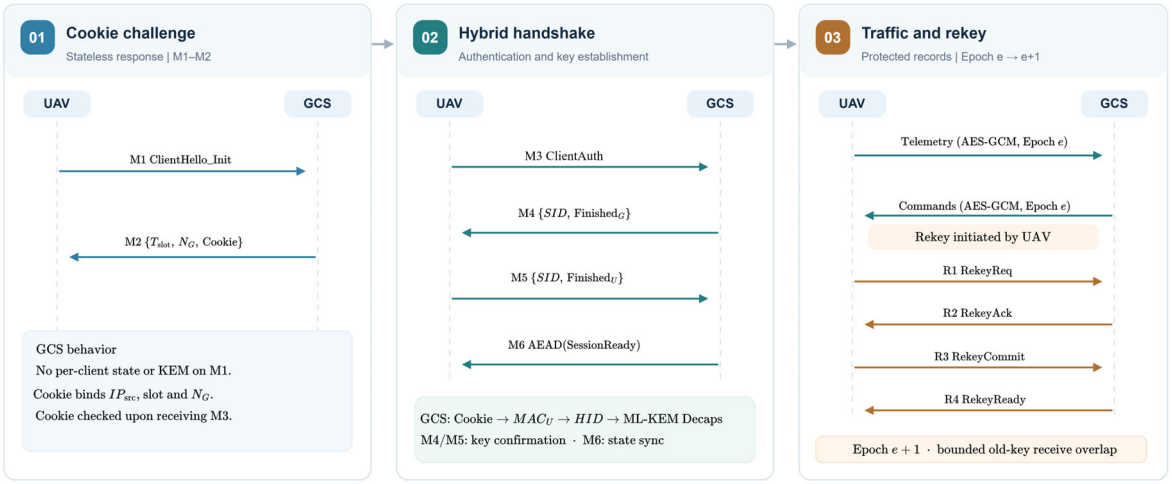


Figure 2. Overall protocol flow.

4.3. Cookie Pre-Authentication and Request Deduplication

The UAV initiates the handshake with M1. The GCS computes the Cookie in Equation (1) using its current time slot T_{slot}, a 128-bit random value N_G, and the source IP address observed on receipt.

$$\text{Cookie} = \text{HMAC}_{\text{MK}}(E(L_C, \text{IP}_{\text{src}}, T_{\text{slot}}, N_G)) \quad (1)$$

The GCS returns T_{slot}, N_G, and Cookie in M2 (CookieReply). It need not store per-request Cookies or N_G values and does not perform ML-KEM decapsulation; the Cookie is recomputed from the same inputs upon receipt of M3.

During the current handshake, the UAV receives M2, reads T_{slot}, N_G, and Cookie, and generates nonce N_U. It then encapsulates under the provisioned pk_G to obtain ciphertext ct and shared secret ss, and computes MAC_U from LTSK_i using Equation (2). The MAC binds both identities, the nonces, Cookie, and ciphertext.

$$\text{MAC}_U = \text{HMAC}_{\text{LTSK}_i}(E(L_A, \text{PID}_i, \text{ID}_G, T_{\text{slot}}, N_U, N_G, \text{Cookie}, \text{ct})) \quad (2)$$

M3 (ClientAuth) contains PID_i, T_{slot}, N_U, N_G, Cookie, ct, and MAC_U. The GCS checks message length, encoding, and time-slot validity, verifies the Cookie against the observed source IP, retrieves LTSK_i using PID_i, and verifies MAC_U, in that order. Failure at any stage causes rejection without decapsulation. Neither ID_G nor the source IP is transmitted in M3.

For an M3 that passes these checks, the GCS derives the handshake identifier HID using Equation (3) to identify duplicate transactions. HID is stored only locally at the GCS.

$$\text{HID} = \text{SHA256}(E(\text{PID}_i, \text{IP}_{\text{src}}^{\text{obs}}, T_{\text{slot}}, N_U, N_G, \text{Cookie}, \text{ct})) \quad (3)$$

The GCS registers a new valid HID as PENDING and performs ML-KEM decapsulation once. If the same HID arrives again while the request is pending, decapsulation is not repeated; if M4 has already been generated, the cached M4 is returned. The GCS limits the number of pending requests and cache retention time to prevent repeated requests from continuously occupying resources. Requests with valid Cookies must still pass MAC_U verification, so reduced computational overhead cannot be guaranteed under all attack conditions.

4.4. Initial Authentication and Session-Key Establishment

4.4.1. Initial Handshake and Mutual Key Confirmation

Figure 3 shows the M1–M6 handshake. Decapsulation occurs only after M3 passes the Cookie, MAC_U, and HID checks; M4 and M5 provide mutual key confirmation, and M6 activates the session.

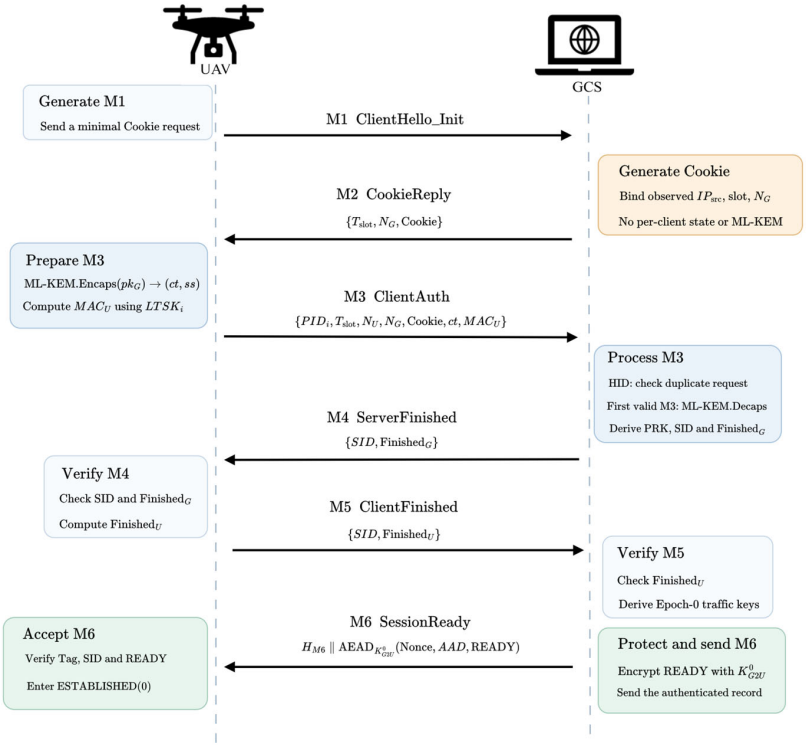


Figure 3. Initial handshake and key confirmation.

Both parties compute the transcript hash TH_3 in Equation (4) from the message types and canonically encoded contents of M1–M3. The network source IP address is excluded from this transcript.

$$TH_3 = \text{SHA256}(E(\text{M1}, \text{M2}, \text{M3})) \tag{4}$$

Using Equations (5) and (6), both parties derive K_{sym} from $LTSK_i$ and use K_{sym} as the HKDF-Extract salt and the ML-KEM shared secret ss as its input to obtain PRK. Thus, the initial key material depends on both types of secrets.

$$K_{sym} = \text{HMAC}_{LTSK_i}(E(L_K, TH_3)) \tag{5}$$

$$\text{PRK} = \text{HKDF-Extract}(K_{sym}, \text{ss}) \tag{6}$$

Equation (7) derives the public session identifier SID, which links encrypted records, replay windows, and subsequent rekeying transactions.

$$\text{SID} = \text{SHA256}(E(L_{SID}, TH_3)) \tag{7}$$

The GCS generates Finished_G using Equations (8) and (9), then sends $\{\text{SID}, \text{Finished}_G\}$ in M4 (ServerFinished). The UAV independently computes and verifies SID and Finished_G before proceeding. Finished_G uses a confirmation key derived from PRK rather than an LTSK-only response MAC.

$$K_{Fin,G} = \text{HKDF-Expand}(\text{PRK}, E(L_{FG}, TH_3), 32) \tag{8}$$

$$\text{Finished}_G = \text{HMAC}_{K_{Fin,G}}(TH_3) \tag{9}$$

After accepting M4, the UAV computes TH_4 using Equation (10) and generates Finished_U with Equations (11) and (12). M5 (ClientFinished) carries $\{\text{SID}, \text{Finished}_U\}$, which the GCS verifies. M4 and M5 employ distinct labels and transcript hashes for confirmation in the two directions.

$$TH_4 = \text{SHA256}(E(TH_3, \text{M4})) \tag{10}$$

$$K_{Fin,U} = \text{HKDF-Expand}(\text{PRK}, E(L_{FU}, TH_4), 32) \tag{11}$$

$$\text{Finished}_U = \text{HMAC}_{K_{Fin,U}}(TH_4) \tag{12}$$

After M5 verification, both parties compute $TH_5 = \text{SHA-256}(\text{E}(TH_4, \text{M5}))$ and derive the initial epoch secret ES_0 using Equation (13).

$$ES_0 = \text{HKDF-Expand}(\text{PRK}, E(L_{ES}, 0, TH_5), 32) \tag{13}$$

Traffic-encryption keys for both directions are derived by Equation (14). Here, j denotes the epoch index; d is U2G or G2U; and L_d denotes the corresponding domain-separation label L_{U2G} or L_{G2U}. Initially, j = 0. Separate labels yield separate direction-specific keys.

$$K_d^j = \text{HKDF-Expand}(ES_j, E(L_d, \text{SID}, j), 32) \tag{14}$$

Figure 4 summarizes the key relationships. $LTSK_i$ and the ML-KEM shared secret jointly determine PRK, from which the Finished confirmation keys and epoch traffic keys are derived. The equations above define the exact encoded inputs.

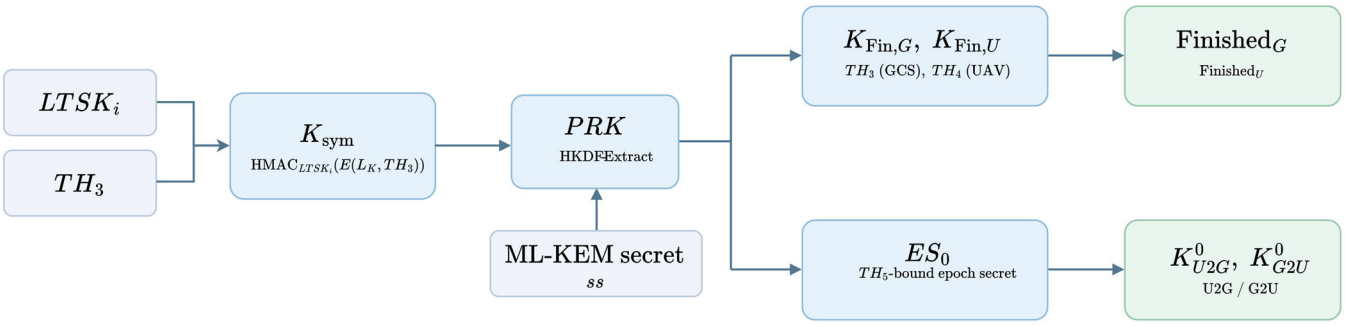


Figure 4. Initial session-key derivation.

After verifying M5, the GCS sends M6 (SessionReady). Its header contains SID, direction G2U, epoch 0, Seq, record type SessionReady, and Length; its encrypted payload is READY. The UAV enters ESTABLISHED only after successfully authenticating the first AES-GCM record. M6 synchronizes session activation; it is not a third Finished exchange.

4.4.2. Encrypted Records, Nonces, and Replay Checks

The encrypted record in Figure 5 comprises Header, Ciphertext, and a 16-byte Tag. The header contains SID, Direction, Epoch, Seq, Type, and Length, where Length denotes the encrypted payload length. The header remains in plaintext, but its canonical encoding is authenticated by AES-GCM as AAD. Telemetry and control commands use K_{U2G} and K_{G2U}, respectively.

Equation (15) constructs a 96-bit nonce from a four-byte Epoch and an eight-byte Seq. Within a given session, direction, and epoch, all record types share the same sequence-number counter. A sender must never reuse a sequence number with the same traffic key. If the sequence-number state cannot be recovered after a restart, a fresh handshake is required.

$$\text{Nonce} = \text{I2OSP}(\text{Epoch}, 4) \parallel \text{I2OSP}(\text{Seq}, 8) \tag{15}$$

The receiver selects exactly one key according to the session state, direction, and epoch. It checks the sequence-number range and potential duplication before verifying the GCM tag. The sequence number is committed to the replay window only after authentication succeeds; invalid records must not advance the window. Separate windows are maintained for each SID, direction, and epoch, with concurrent acceptance of the same Seq prevented.

Figure 5 illustrates direction-separated keys and the requirement to update the replay window only after successful authentication. Old-epoch records may be received only during the transition specified in Section 4.5.

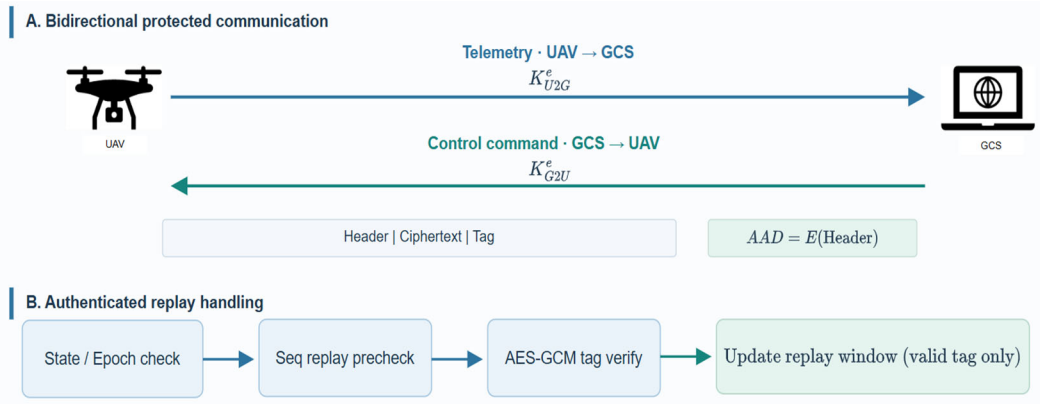


Figure 5. Bidirectional record protection and replay checking.

4.5. Key Updates and Error Handling

4.5.1. Rekeying Message Exchange

Let e be the active traffic epoch and e′ = e + 1 the proposed next epoch. Only one rekeying transaction may be pending for a given SID. The UAV generates a new RID and ML-KEM material ct′ and ss′, and computes the rekeying transcript hash RH in Equation (16), binding the SID, both epoch numbers, and ciphertext.

$$\mathrm{RH} = \mathrm{SHA256}(E(L_R, \mathrm{SID}, e, e', \mathrm{RID}, \mathrm{ct}')) \tag{16}$$

The peers combine the old epoch secret $\mathrm{ES_e}$ and ss′ to obtain RPRK′ using Equation (17), then derive $\mathrm{ES_e}'$ using Equation (18). The new traffic keys follow Equation (14) with j = e′. Equations (19) and (20) specify the rekeying confirmation key $\mathrm{K_{RK}}$ and the GCS confirmation value $\mathrm{K_{CG}}$.

$$\mathrm{RPRK}' = \mathrm{HKDF\text{-}Extract}(\mathrm{ES}_e, \mathrm{ss}') \tag{17}$$

$$\mathrm{ES}_{e'} = \mathrm{HKDF\text{-}Expand}(\mathrm{RPRK}', E(L_{\mathrm{ES}}, e', \mathrm{RH}), 32) \tag{18}$$

$$K_{\mathrm{RK}} = \mathrm{HKDF\text{-}Expand}(\mathrm{ES}_{e'}, E(L_{\mathrm{RK}}, \mathrm{RH}), 32) \tag{19}$$

$$\mathrm{KC}_G = \mathrm{HMAC}_{K_{\mathrm{RK}}}(E(\mathrm{RH}, \mathrm{RID}, e')) \tag{20}$$

The UAV sends R1 (RekeyReq) under the old uplink key. Its header identifies U2G, epoch e, and type RekeyReq; the payload is {e′, RID, ct′}. After authenticating the record and validating the transaction fields, the GCS decapsulates ct′ only for the first valid RID, retains candidate keys, and enters PREPARED.

The GCS sends R2 (RekeyAck) under the old downlink key, with G2U, epoch e, and record type RekeyAck in the header and {RID, e′, $\mathrm{K_{CG}}$} in the payload. The UAV sends R3 under the new epoch only after verifying $\mathrm{K_{CG}}$. At this point, neither party has switched its data-sending key.

R3 (RekeyCommit) is protected with the new uplink key, specifies U2G and epoch e′, and carries {RID, COMMIT}. After its first successful verification, the GCS enters COMMITTED, switches its control-command sending key to e′, and sends R4 (RekeyReady) with the new downlink key and payload {RID, READY}. The UAV switches its telemetry sending key to e′ only after authenticating R4, completing the update.

Figure 6 shows that R1 and R2 are protected under the old-epoch keys, whereas R3 and R4 use the new-epoch keys. The GCS switches its command-transmission key after validating R3, while the UAV switches its telemetry-transmission key after validating R4.

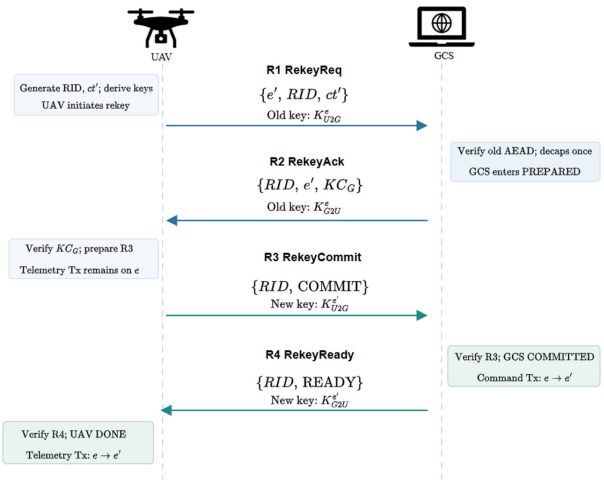


Figure 6. Key-update procedure.

4.5.2. Retransmission and Key-Reception Rules

If R1 or R2 is lost, the UAV retransmits the logical R1 with the same RID, ct′, and e′, using a fresh old-epoch Seq. After validating the duplicate request, the GCS reuses the stored transaction information and re-encrypts R2 with a fresh Seq, without repeating ML-KEM decapsulation. A request with the same RID but a different ciphertext or target epoch is rejected. Retransmissions are bounded by RTO and a retry limit.

If R3 or R4 is lost, the UAV retransmits the logical R3 with the same RID and COMMIT, using a fresh new-epoch Seq. After verifying that the RID matches the committed transaction, the GCS re-encrypts R4 with a fresh Seq without repeating the COMMITTED transition. Each retransmission uses a newly generated Nonce and Tag. Retry exhaustion is handled as described in Section 4.5.3.

The receiver selects the key according to transaction state, direction, and Epoch in the header. The GCS can validate a new-epoch R3 in PREPARED and may temporarily accept state-valid old-epoch telemetry after entering COMMITTED. Overlapping reception is time-bounded, with a separate replay window for each epoch; the receiver does not try both keys against a single record.

4.5.3. Timeouts and Key Deletion

When retries are exhausted, the GCS may abandon the pending update and retain the old epoch if it remains in PREPARED and has received no authenticated evidence that the peer uses the new epoch. Once COMMITTED, it must not revert directly; prolonged desynchronization requires terminating SID and performing a fresh handshake. Terminating reception of the old epoch additionally depends on authenticated evidence of peer use of the new epoch and the grace period T_{grace}. Thus, uninterrupted operation cannot be guaranteed under persistent acknowledgment loss. After the exit conditions are met, the old-epoch secret, traffic keys, and replay window are deleted, and transaction records are purged according to their retention periods. Unneeded ss, K_{sym}, and Finished-derived keys are also erased after the handshake. An expired Cookie or initial-handshake timeout requires restarting from M1.

Figure 7 illustrates retransmissions after R2 or R4 loss and the treatment of PREPARED and COMMITTED states. Old-epoch reception remains subject to transaction-state and timeout restrictions.

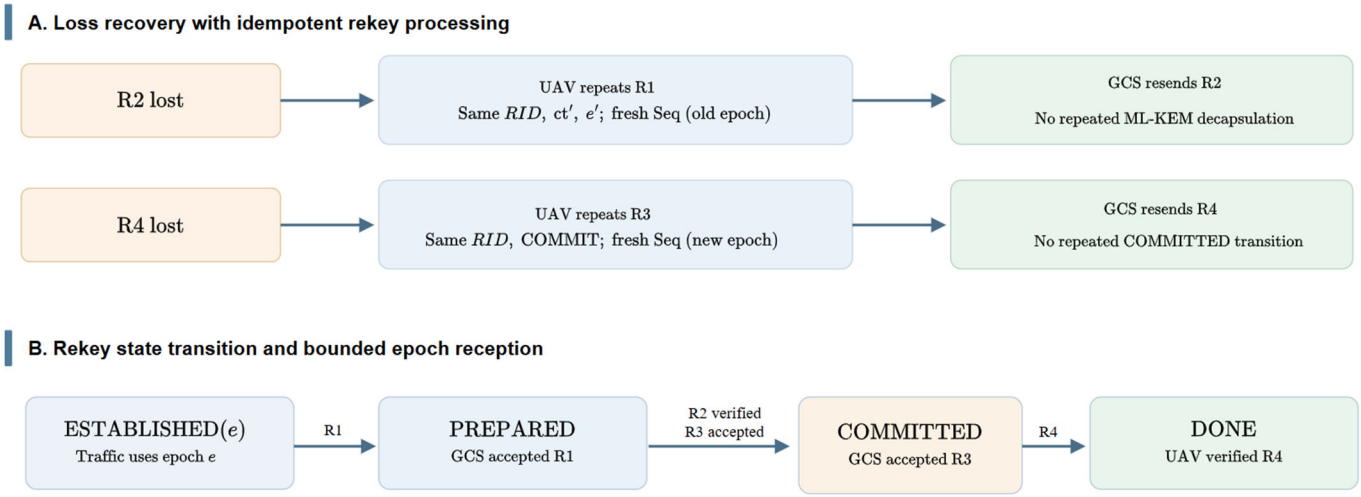


Figure 7. Recovery from key-update failures.

5. Security Analysis and Formal Verification

This section uses ProVerif 2.05 to formally verify QUAD. It examines confidentiality and authentication properties under initial-handshake, rekeying, duplicate-message, and long-term secret disclosure scenarios, and specifies the scope of the resulting claims.

5.1. ProVerif Modeling of Protocol Operations

Following the message flows and key computations in Section 4, separate ProVerif 2.05 models are constructed for the initial handshake, normal rekeying, bounded duplicate handling, and long-term secret disclosure. Secrecy and event-correspondence queries are used to assess the protocol properties. The rekeying models assume an already established secure session. Each model is executed independently rather than representing a continuous execution from the initial handshake through multiple key updates. The queries and results are presented in Sections 5.2 and 5.3.

The model uses the Dolev–Yao adversary defined in Section 3.2, who controls the public channel and can initiate concurrent sessions. SHA-256, HMAC, HKDF, ML-KEM, and AES-GCM are abstracted as ideal cryptographic primitives; constructors, destructors, and equations represent the permitted computations. LTSKs, the GCS private key, and the Cookie key remain private unless explicitly disclosed. Accordingly, the findings concern protocol properties within the symbolic model and do not constitute computational-security proofs of the underlying algorithms.

The initial-handshake model distinguishes message reception from successful authentication. Source IP addresses observed by the GCS for M1 and M3 are modeled as external parameters for Cookie generation and validation, and the observed M3 source IP also contributes to HID. The GCS verifies the Cookie and MAC_U in sequence before proceeding. A repeated HID either receives the cached response or waits for the initial processing, without repeating decapsulation. The GCS_Accept_UAV event is placed after successful MAC_U verification for correspondence checking. Figure 8 shows the principal checks and event locations.

```
(a) GCS: M3 authentication
if cookie = calc_cookie(mk_cookie,ip_src_m3,t_slot,n_g) then
get ltsk_table(=u_req,k_ltsk:sym_key) in
if mac_u = hmac(k_ltsk,macu_input(u_req,g,t_slot,n_u,n_g,cookie,ct)) then
let auth_id = make_auth_id(u_req,t_slot,n_u,n_g,cookie,ct) in
let hid = make_hid(u_req,ip_src_m3,t_slot,n_u,n_g,cookie,ct) in
event GCS_Accept_UAV(g,u_req,auth_id);
get hs_cache(=hid,sid_cached:sid_t,fin_cached:mac_t) in
out(c,(ServerFinished,sid_cached,fin_cached))
else
get hs_claim(=hid) in 0 else
insert hs_claim(hid);
let ss_g = mlkem_decaps(sk_gcs,ct) in
(b) UAV: M4 verification and M5 confirmation
if sid_recv = sid_u then
let k_fin_g_u = hkdf_expand(prk_u,info_fin_g(th3_u)) in
if fin_g_recv = hmac(k_fin_g_u,hash2bits(th3_u)) then
event UAV_Accept_GCS_Finished(g,u,sid_u,prk_u);
let th4_u = make_th4(th3_u,canon_m4(sid_u,fin_g_recv)) in
let k_fin_u = hkdf_expand(prk_u,info_fin_u(th4_u)) in
let fin_u = hmac(k_fin_u,hash2bits(th4_u)) in
event UAV_Send_Finished(g,u,sid_u,prk_u);
```

Figure 8. Initial handshake authentication and key confirmation.

Figure 8a illustrates the checks preceding decapsulation and three HID cases: first processing, pending processing, and cache hit. The model uses table operations to prevent repeated decapsulation of the same transaction, but does not simulate actual Cookie expiration checks, cache capacity limits, or concurrent request handling. These rules are specified in Section 4 and evaluated where applicable in Section 6.

M4 and M5 provide key confirmation in the two directions. After the UAV verifies SID and Finished_G computed from a PRK-derived key, it triggers UAV_Accept_GCS_Finished and generates Finished_U. The GCS triggers GCS_Accept_UAV_Finished only after verifying Finished_U. Both Finished messages therefore depend on keys derived in the current handshake rather than solely on an LTSK-based MAC.

The GCS protects the READY payload of M6 using the initial-epoch downlink key. The UAV triggers UAV_Established only after AES-GCM authentication and READY verification. Q3–Q6 check event correspondences for M3 authentication, M4/M5 key confirmation, and M6 session establishment; separate queries check bidirectional data confidentiality.

The rekeying model uses SID for the session and RID for the update transaction, binding the epoch and ciphertext to the rekeying transcript. After validating R1, the GCS performs the initial decapsulation and prepares a new key. The UAV verifies R2 and sends new-epoch R3; the GCS commits the update after validating R3, and the UAV completes its transition after validating R4. Events are placed at these stages for correspondence checks Q9–Q11.

The duplicate-message model distinguishes the first processing of a transaction from subsequent processing with the same RID. A repeated R1 must be authenticated and match the original RID, target epoch, and ciphertext; it only triggers retransmission of R2. A repeated R3 must pass new-epoch authentication and match the RID; it only triggers retransmission of R4, without recommitting the state. Figures 9 and 10 show the initial R1 authentication/decapsulation and duplicate R1/R3 acknowledgment handling, respectively.

```
let p1 = aead_dec(k_u2g_old,n1,ad1,e1) in
if get_req_new(p1) = Epoch1 then
let rid = get_req_rid(p1) in let ct = get_req_ct(p1) in
let ss = mlkem_decaps(sk_gcs,ct) in
let rh = make_rh(sid,Epoch0,Epoch1,rid,ct) in
let rprk = hkdf_extract_rekey(es0,ss) in
let es1 = hkdf_expand(rprk,info_es(Epoch1,rh)) in
let ku1 = hkdf_expand(es1,info_u2g(sid,Epoch1)) in
let kg1 = hkdf_expand(es1,info_g2u(sid,Epoch1)) in
let krk = hkdf_expand(es1,info_rekey_confirm(rh)) in
let kc = hmac(krk,kc_input(rh,rid,Epoch1)) in
event GCS_Rekey_Decap_Once(g,u,sid,rid,es1);
```

Figure 9. R1 authentication and decapsulation.

```
(a) Duplicate R1: acknowledgment only
let p2 = aead_dec(k_u2g_old,n2,ad2,e2) in
if get_req_new(p2) = Epoch1 then
if get_req_rid(p2) = rid then
if get_req_ct(p2) = ct then
event GCS_Duplicate_Req_AckOnly(g,u,sid,rid,es1);
(b) First R3 and duplicate R3
(processFirstR3: first valid R3)
let p_c1 = aead_dec(ku1,n_c1,ad_c1,enc_c1) in
if get_commit_rid(p_c1) = rid then
event GCS_Rekey_Commit_Transition(g,u,sid,rid,es1);
(processOptionalR3: after first commit)
let p_c2 = aead_dec(ku1,n_c2,ad_c2,enc_c2) in
if get_commit_rid(p_c2) = rid then
event GCS_Duplicate_Commit_ReadyOnly(g,u,sid,rid,es1);
```

Figure 10. Confirmation handling for R1 and R3.

The events GCS_Rekey_Decap_Once and GCS_Rekey_Commit_Transition mark the first decapsulation and commitment, while GCS_Duplicate_Req_AckOnly and GCS_Duplicate_Commit_ReadyOnly mark duplicate acknowledgment handling. Q12–Q16 check the corresponding event relationships, with fresh Seq values used for retransmitted records. The model covers only one additional R1 and R3 path and does not fully model arbitrary retransmissions, overlapping reception of old- and new-epoch traffic, replay windows, or old-key deletion.

In the single-secret disclosure models, an initial handshake is first completed and the UAV triggers M6Authenticated after authenticating M6 and READY. Historical ciphertexts are then generated in both directions and made public, triggering HistoricalRecordsEmitted. Only afterward does DisclosureAfterM6 occur and one long-term secret becomes available to the adversary. Separate models disclose the LTSK or GCS ML-KEM private key, and Q17–Q22 assess historical data confidentiality and event ordering.

The predefined attack models include two classes. Q23 and Q25 disclose both the LTSK and the GCS ML-KEM private key after M6 authentication and historical ciphertext generation, checking historical telemetry and control-command confidentiality. Q24 and Q26 disclose the target UAV's LTSK before a future handshake to examine impersonation risk. Sections 5.2 and 5.3 present their queries and results.

5.2. Security Events and Verification Queries

ProVerif checks the ordering of protocol actions with event-correspondence queries. A ==> B requires a matching event B to have occurred before A; inj-event additionally requires distinct instances of the queried event to correspond to distinct preceding instances. These events are not transmitted as protocol messages. Table 2 defines the main events, and independent models do not share runtime state.

Table 2. Authentication and key-update events.

Event Category	Event Names	Protocol Action Represented
M3 authentication	UAV_Auth_Request; GCS_Accept_UAV	UAV request initiation and successful M3 validation by the GCS
M4 key confirmation	GCS_Send_Finished; UAV_Accept_GCS_Finished	GCS sends Finished; UAV verifies and accepts it
M5 key confirmation	UAV_Send_Finished; GCS_Accept_UAV_Finished	UAV sends Finished; GCS verifies it
M6 session activation	GCS_Send_SessionReady; UAV_Established	GCS sends M6; UAV establishes the session after authenticated READY
Normal rekeying	UAV_Rekey_Request; GCS_Rekey_Prepared; UAV_Rekey_Commit; GCS_Rekey_Commit; GCS_Rekey_Ready; UAV_Rekey_Done	R1 request, GCS preparation, R3 commit, R4 readiness, and UAV completion
Duplicate R1/R3	GCS_Rekey_Decap_Once; GCS_Duplicate_Req_AckOnly; UAV_Rekey_Commit_Start; GCS_Rekey_Commit_Transition; GCS_Duplicate_Commit_ReadyOnly	Distinguishes initial cryptographic processing from confirmation-only retransmission
Disclosure chronology	M6Authenticated; HistoricalRecordsEmitted; DisclosureAfterM6	M6 authentication, historical ciphertext emission, then single-secret disclosure
Predefined impersonation	LTSKCompromisedBeforeFutureHandshake; GCS_Future_Accept_UAV; Honest_UAV_Future_Request	Checks correspondence between future GCS acceptance and an honest UAV request
Predefined dual disclosure	PastSessionBeforeDualCompromise	After the historical ciphertexts are generated, prepare to disclose both long-term secrets

A total of 26 queries are defined: Q1–Q6 assess initial-handshake secrecy and authentication; Q7–Q11 assess rekeying; Q12–Q16 assess bounded duplicate handling; Q17–Q22 address single-secret disclosure after M6; and Q23–Q26 address predefined dual disclosure and future impersonation. True means that the queried property holds in the corresponding model, whereas False indicates an expected counterexample under the stated attack conditions. For secrecy queries, 'not attacker(secret) is true' means that the adversary cannot derive the test secret. Table 3 lists all queries and results.

Table 3. ProVerif query results (Q1–Q26).

ID	Original ProVerif Query	Result
Initial handshake secrecy and authentication (Q1–Q6)		
Q1	query attacker(secret_u2g).	True
Q2	query attacker(secret_g2u).	True
Q3	query g:host,u:host,h:hash_t; event(GCS_Accept_UAV(g,u,h)) ==> event(UAV_Auth_Request(g,u,h)).	True
Q4	query g:host,u:host,s:sid_t,p:sym_key; inj-event(UAV_Accept_GCS_Finished(g,u,s,p)) ==> inj-event(GCS_Send_Finished(g,u,s,p)).	True
Q5	query g:host,u:host,s:sid_t,p:sym_key; event(GCS_Accept_UAV_Finished(g,u,s,p)) ==> event(UAV_Send_Finished(g,u,s,p)).	True
Q6	query g:host,u:host,s:sid_t,k:sym_key; inj-event(UAV_Established(g,u,s,k)) ==> inj-event(GCS_Send_SessionReady(g,u,s,k)).	True

Table 3. *Cont.*

ID	Original ProVerif Query	Result
Rekeying secrecy and event correspondence (Q7–Q11)		
Q7	query attacker(secret_rekey_u2g).	True
Q8	query attacker(secret_rekey_g2u).	True
Q9	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; event(GCS_Rekey_Prepared(g,u,s,r,e)) ==> event(UAV_Rekey_Request(g,u,s,r,e)).	True
Q10	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; event(GCS_Rekey_Commit(g,u,s,r,e)) ==> event(UAV_Rekey_Commit(g,u,s,r,e)).	True
Q11	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; inj-event(UAV_Rekey_Done(g,u,s,r,e)) ==> inj-event(GCS_Rekey_Ready(g,u,s,r,e)).	True
Bounded duplicate requests and confirmations (Q12–Q16)		
Q12	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; inj-event(GCS_Rekey_Decap_Once(g,u,s,r,e)) ==> inj-event(UAV_Rekey_Request(g,u,s,r,e)).	True
Q13	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; event(GCS_Duplicate_Req_AckOnly(g,u,s,r,e)) ==> event(GCS_Rekey_Decap_Once(g,u,s,r,e)).	True
Q14	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; inj-event(GCS_Rekey_Commit_Transition(g,u,s,r,e)) ==> inj-event(UAV_Rekey_Commit_Start(g,u,s,r,e)).	True
Q15	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; event(GCS_Duplicate_Commit_ReadyOnly(g,u,s,r,e)) ==> event(GCS_Rekey_Commit_Transition(g,u,s,r,e)).	True
Q16	query g:host,u:host,s:sid_t,r:rid_t,e:sym_key; event(UAV_Rekey_Done(g,u,s,r,e)) ==> event(GCS_Rekey_Commit_Transition(g,u,s,r,e)).	True
Single-secret disclosure after M6 (Q17–Q22)		
Q17	query attacker(secret_ltsk_broken).	True
Q18	query attacker(secret_ltsk_broken_g2u).	True
Q19	query attacker(secret_mlkem_broken).	True
Q20	query attacker(secret_mlkem_broken_g2u).	True
Q21	query u:host,s:sid_t; event(DisclosureAfterM6(u,s)) ==> event(M6Authenticated(u,s)).	True
Q22	query u:host,s:sid_t; event(DisclosureAfterM6(u,s)) ==> event(HistoricalRecordsEmitted(u,s)).	True
Predefined dual disclosure and future impersonation (Q23–Q26)		
Q23	query attacker(secret_both_broken).	False
Q24	query attacker(future_impersonation_witness).	False
Q25	query attacker(secret_both_broken_g2u).	False
Q26	query u:host; event(GCS_Future_Accept_UAV(u)) ==> event(Honest_UAV_Future_Request(u)).	False

5.3. Verification Results

Formal verification of QUAD was conducted using ProVerif 2.05, yielding 26 query results, of which 22 returned True and four returned False under predefined attack condi-

tions. The complete query results are presented in Table 3 and analyzed below according to their verification objectives.

Queries Q1–Q6 all returned True. Q1 and Q2 confirm that the test secrets for both communication directions in the initial session are not derivable by the attacker. Q3–Q6 establish the corresponding event relationships for M3 authentication, M4/M5 key confirmation, and M6 session establishment. Specifically, Q4 and Q6 use injective correspondence queries, whereas Q3 and Q5 use non-injective queries. Therefore, not all authentication results imply a unique one-to-one correspondence. These results support the initial-handshake confidentiality, mutual authentication, and key confirmation properties of QUAD within the modeled conditions.

Queries Q7–Q11 all returned True, indicating that the test secrets in both directions remain confidential after rekeying and that the preparation, commitment, and completion events satisfy their specified correspondence relationships. Q12–Q16 also returned True, further verifying the event constraints associated with the first R1 decapsulation, duplicate R1 acknowledgment retransmission, first R3 commitment, and duplicate R3 acknowledgment retransmission. These results support the modeled design for avoiding repeated execution of critical operations within the same transaction. However, the model covers only the specified paths involving one additional R1 and R3, rather than arbitrary retransmissions or complex message interleavings.

Queries Q17–Q22 all returned True. In particular, Q17–Q20 show that the test secrets for both directions in the specified historical sessions remain non-derivable when either the LTSK or the GCS ML-KEM private key is disclosed individually after M6 authentication and historical ciphertext generation. Q21 and Q22 further verify the ordering relationships between the disclosure events and preceding session events. In contrast, Q23–Q26 all returned False, representing expected counterexamples under predefined compromise conditions. Q23 and Q25 show that historical data can be recovered when both long-term secrets are disclosed, while Q24 and Q26 reveal the risk of future UAV impersonation following LTSK disclosure.

Overall, QUAD satisfies the modeled initial-handshake confidentiality, authentication, key confirmation, and rekeying event correspondence properties. Historical test secrets remain non-derivable under the specified single-secret disclosure conditions after M6, while the dual-disclosure and future-impersonation models produce expected counterexamples. These results do not establish full forward secrecy or protocol correctness under arbitrary network conditions. Practical retransmissions, cache management, and key switching are further evaluated in Section 6.

6. Experimental Results and Analysis

6.1. Experimental Setup and Metrics

Experiments were conducted on Raspberry Pi Zero 2 W and PC platforms to assess cryptographic operations, full handshakes, data authentication, abnormal request processing, and rekeying. Normally, the Raspberry Pi emulated the UAV and the PC emulated the GCS; in stress tests, the Raspberry Pi instead served as the GCS responder. The prototype uses ML-KEM-768, HMAC-SHA-256, HKDF-SHA-256, and AES-256-GCM. All experimental categories were run independently.

Cryptographic-operation benchmarks and full-handshake tests were each conducted in five independent runs. The full-handshake tests comprised 1000 handshake attempts per run. Stress tests used three independent runs per configuration, each containing 10 legitimate handshakes. Metrics include cryptographic-operation time, process CPU time, end-to-end handshake latency, and data-reception rates. Communication overhead is recorded as UDP application payload, IPv4 datagram length, and captured frame length.

6.2. Cryptographic and Handshake Overhead

Execution times of cryptographic operations were measured separately at the UAV and GCS. SHA-256 and HMAC-SHA-256 used baseline input lengths and actual protocol-message lengths; AES-256-GCM used a 5 B READY payload and 52 B of additional authenticated data (AAD). Each test had five independent runs, as summarized in Table 4.

Table 4. Execution time of individual cryptographic operations.

Cryptographic Operation	Input Size or Parameters	UAV (μs)	GCS (μs)
SHA-256 hash	32 B	7.528 ± 0.007	0.595 ± 0.026
HMAC-SHA-256	32 B	28.312 ± 0.121	2.081 ± 0.189
Cookie HMAC	58 B	29.345 ± 0.120	2.149 ± 0.107
MAC_U HMAC	1237 B	45.497 ± 0.094	4.448 ± 0.094
HID SHA-256	1208 B	24.852 ± 0.016	3.095 ± 0.178
TH_3 SHA-256	1279 B	25.722 ± 0.009	3.171 ± 0.041
ML-KEM-768 encapsulation	ML-KEM-768	1344.432 ± 0.798	14.684 ± 0.630
ML-KEM-768 decapsulation	ML-KEM-768	1650.851 ± 0.213	16.892 ± 1.067
AES-256-GCM encryption	5 B payload, 52 B AAD	16.189 ± 0.096	0.935 ± 0.021

Table 4 shows that ML-KEM-768 encapsulation on the Raspberry Pi requires 1344.432 μs, substantially longer than the HMAC computations for Cookie and MAC_U, whereas encapsulation on the PC requires only 14.684 μs. Based on the cryptographic-operation counts, initial-handshake computation times are estimated at approximately 1.7148 ms for the UAV and 0.0535 ms for the GCS, excluding communication and program waiting time.

A wolfSSL-based DTLS 1.3 PSK–P-256 scheme and an in-house PSK–ML-KEM-768 scheme were selected as baselines. Each protocol was tested independently in five runs of 1000 handshakes, measuring end-to-end latency, process CPU time at both endpoints, and communication overhead. Results are shown in Table 5.

Table 5. Handshake performance comparison.

Protocol	Successful/ Attempted	UAV Handshake Latency (ms)	UAV Process CPU (ms/Handshake)	GCS Process CPU (ms/Handshake)	UAV Handshake P95 (ms)	UDP Handshake Payload (B)
QUAD (this work)	5000/5000	51.077 ± 2.066	3.454 ± 0.008	0.422 ± 0.090	65.246	1466
DTLS 1.3 (PSK + P-256)	5000/5000	44.753 ± 3.525	4.899 ± 0.021	0.875 ± 0.048	54.746	1119
PSK + ML-KEM-768 (in-house baseline)	5000/5000	47.937 ± 1.063	2.921 ± 0.027	0.350 ± 0.095	64.582	1368

Table 5 shows mean handshake latencies of 51.077 ms for QUAD, 44.753 ms for the DTLS reference, and 47.937 ms for the in-house baseline. Relative to the latter, QUAD increased UAV process CPU time by 0.533 ms per handshake and normal-handshake UDP payload by 98 bytes, primarily because of the additional Cookie exchange and request checks. QUAD used less process CPU time at both endpoints than DTLS but transmitted more bytes. Because DTLS uses classical elliptic-curve key exchange and the in-house baseline omits Cookie pre-authentication, these measurements reflect different key-exchange and authentication procedures and must be interpreted alongside their respective security mechanisms.

6.3. Communication Overhead

Normal, retransmission-free QUAD and DTLS 1.3 handshakes were captured using Wireshark. The number of UDP datagrams, total application payload, and IPv4 datagram lengths were compared, as illustrated in Figure 11.

(a)

No.	Time	Source	Destination	Protocol	Lengt	Total Length	Info
1	19:56:38.296932	192.168.237.99	192.168.237.149	UDP	49	29	49373 → 44444 Len=1
2	19:56:38.306543	192.168.237.149	192.168.237.99	UDP	105	85	44444 → 49373 Len=57
3	19:56:38.332479	192.168.237.99	192.168.237.149	UDP	1257	1237	49373 → 44444 Len=1209
4	19:56:38.342967	192.168.237.149	192.168.237.99	UDP	113	93	44444 → 49373 Len=65
5	19:56:38.343694	192.168.237.99	192.168.237.149	UDP	113	93	49373 → 44444 Len=65
6	19:56:38.349856	192.168.237.149	192.168.237.99	UDP	117	97	44444 → 49373 Len=69

(b)

No.	Time	Source	Destination	Protocol	Length	Total Length	Info
1	19:57:12.013229	192.168.237.99	192.168.237.149	DTLSv1.3	320	300	Client Hello
2	19:57:12.053411	192.168.237.149	192.168.237.99	DTLSv1.3	192	172	Hello Retry Request
3	19:57:12.053958	192.168.237.99	192.168.237.149	DTLSv1.3	393	373	Client Hello
4	19:57:12.061680	192.168.237.149	192.168.237.99	DTLSv1.3	198	178	Server Hello
5	19:57:12.063286	192.168.237.149	192.168.237.99	DTLSv1.3	84	64	Encrypted Data
6	19:57:12.063410	192.168.237.149	192.168.237.99	DTLSv1.3	114	94	Encrypted Data
7	19:57:12.064911	192.168.237.99	192.168.237.149	DTLSv1.3	114	94	Encrypted Data
8	19:57:12.071703	192.168.237.149	192.168.237.99	DTLSv1.3	88	68	Encrypted Data

Figure 11. Captured handshake packets: (**a**) QUAD; (**b**) DTLS 1.3.

Figure 11 shows that a QUAD handshake exchanges six UDP datagrams with a total application payload of 1466 bytes. M3 accounts for 1209 bytes, largely comprising the ML-KEM-768 ciphertext and authentication data. The DTLS 1.3 reference exchanged eight datagrams totaling 1119 payload bytes. QUAD therefore used two fewer datagrams but 347 more payload bytes (approximately 31.0%). Including IP and UDP headers, the respective IPv4 datagram totals were 1634 and 1343 bytes. These figures characterize application- and network-layer costs during normal handshakes; they exclude wireless link-layer retransmissions and physical-layer overhead.

6.4. Data Authentication and Replay Protection

Authenticated-record processing was tested with pre-provisioned session keys for simulated UAV telemetry and GCS commands. Scenarios included valid in-order and out-of-order delivery, ciphertext and tag tampering, incorrect direction keys, duplicate records, and invalid headers. A sliding replay window of width 64 was used over five independent runs; Table 6 reports the outcomes.

Table 6. AES-GCM authentication and replay test results.

Test Case	Record Count	Expected Outcome	Observed Outcome
Normal and valid out-of-order records	70,010	Accepted	70,010 correctly accepted
Ciphertext/tag/AAD tampering or incorrect direction key	40,000	Authentication failure and rejection	40,000 correctly rejected
Duplicates and out-of-order replay	20,000	Replay rejection	20,000 correctly rejected
Old records outside replay window	10	Too-old rejection	10 correctly rejected
Invalid SID, direction, epoch, type, or length	50	Format/state rejection	50 correctly rejected
Total functional test	130,070	As expected	130,070 outcomes as expected
GCS uplink validation latency	5000 samples	—	10.067 ± 0.551 µs
UAV downlink validation latency	5000 samples	—	69.446 ± 1.615 µs

As shown in Table 6, all 70,010 legitimate records were accepted and all 60,060 abnormal records were correctly rejected, without improperly updating the replay window. Mean normal-record validation times at the GCS and UAV were 10.067 ± 0.551 µs and 69.446 ± 1.615 µs, respectively, demonstrating bidirectional data authentication and replay checks under the tested conditions.

6.5. Computational Cost Under Invalid Requests

The Raspberry Pi acted as the GCS responder in stress tests comprising M1 flooding, invalid Cookies, valid Cookies with incorrect MACs, and duplicate legitimate M3 packets. Full QUAD was compared with an HMAC-only checking mode. Each condition was repeated three times, with ten legitimate handshakes per run. Process CPU time, request-processing latency, rejections, and legitimate-handshake performance are presented in Table 7.

Table 7. Request processing under Raspberry Pi GCS stress.

(**a**) Invalid-request tests

Test scenario	Rate (pps)	CPU (ms)	M3 (µs)	Rejects/dedup.	RSS (KiB)
Full QUAD/no attack	0	46.83 ± 0.02	—	—	5612
Full QUAD/M1 flooding	500.02	639.81 ± 10.95	—	—	5612
Full QUAD/invalid Cookie	499.86	453.01 ± 9.43	68.42	11,669 Cookie rejections	5612
HMAC-only/invalid Cookie	499.52	542.20 ± 18.71	93.60	11,630 HMAC rejections	5612
Full QUAD/valid Cookie, wrong MAC	469.73	438.66 ± 70.39	159.00	6474 HMAC rejections	5612
HMAC-only/valid Cookie, wrong MAC	498.79	384.85 ± 26.10	107.79	7411 HMAC rejections	5612
Full QUAD/duplicate valid M3	480.65	682.37 ± 66.91	232.49	11,547 HID deduplications	5612

(**b**) Legitimate-handshake tests

Test scenario	Successful handshakes	Latency (ms)	Decapsulations
Full QUAD/no attack	30/30	33.96 ± 5.10	30
Full QUAD/M1 flooding	30/30	33.88 ± 6.05	30
Full QUAD/invalid Cookie	30/30	32.28 ± 1.14	30
HMAC-only/invalid Cookie	30/30	31.90 ± 2.91	30
Full QUAD/valid Cookie, wrong MAC	30/30	77.34 ± 24.98	30
HMAC-only/valid Cookie, wrong MAC	30/30	106.52 ± 82.93	30
Full QUAD/duplicate valid M3	30/30	76.85 ± 8.17	30

Under invalid-Cookie traffic at approximately 500 pps, full QUAD and the HMAC-only mode consumed 453.01 and 542.20 ms of CPU time, respectively, a reduction of approximately 16.4% for full QUAD. Their mean M3 handling times were 68.42 and 93.60 µs. This indicates that an early Cookie check can reduce authentication costs for invalid requests; Figure 12 shows CPU overhead at different attack rates.

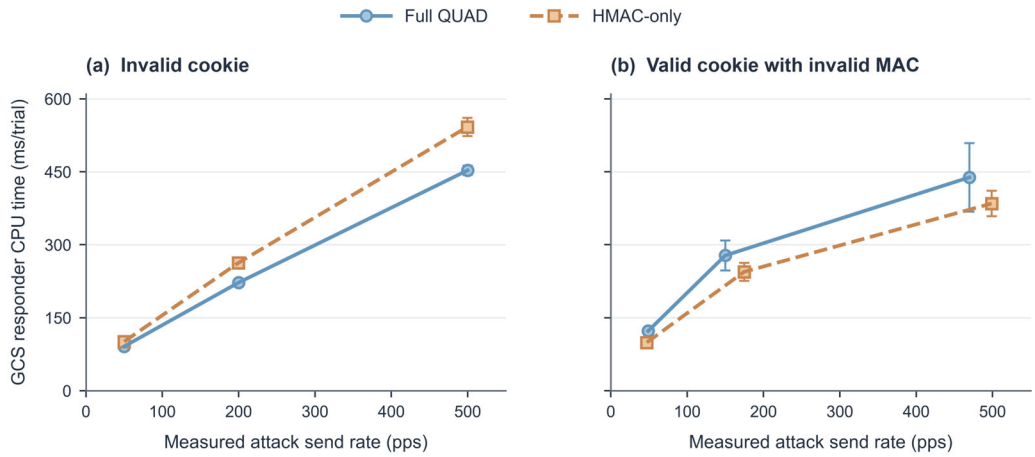


Figure 12. GCS CPU cost under Cookie attacks.

Under invalid-Cookie traffic, full QUAD consistently consumed less process CPU time than HMAC-only checking, with costs rising as attack rates increased (Figure 12). Under valid-Cookie, incorrect-MAC traffic, however, full QUAD consumed 438.66 ms at 469.73 packets/s, compared with 384.85 ms for HMAC-only checking at 498.79 packets/s. Such requests pass the Cookie check and still require MAC verification, adding an additional check in full QUAD. Because the realized attack rates differ, the CPU-time difference cannot be attributed exclusively to Cookie verification.

The M1-flooding condition consumed 639.81 ms of GCS process CPU time. For duplicate legitimate M3 requests, the HID cache detected 11,547 duplicates. All legitimate handshakes completed in the tested groups. These results indicate that Cookie pre-authentication reduces the cost of invalid-Cookie traffic and HID caching avoids repeated expensive decapsulation for matching handshake transactions. Valid-Cookie traffic can nevertheless incur additional verification costs.

6.6. Key Updates and Failure Recovery

A Raspberry Pi Zero 2 W and a PC emulated the UAV and GCS, respectively, while UDP fault injection produced losses, duplicates, out-of-order packets, and repeated rekeying across R1–R4. Rekeying success, recovery time, and bidirectional data reception were recorded. The retransmission timeout (RTO) was 200 ms, with at most five retransmissions; the bidirectional data interval was 20 ms, and the old/new-epoch transition period was 500 ms. Conventional tests lasted 5 s per run with one intended update, and each deterministic fault scenario was repeated five times. Continuous updates and communication outages followed their respective separate configurations. Table 8 presents the results.

In Table 8, first loss means that the initial transmission of a specified R1–R4 message fails, whereas consecutive loss means that R2 or R4 fails twice in succession. Duplicate tests resend a message belonging to the same transaction. R2 reordering and a late old R2 were implemented with delays of 450 and 1200 ms, respectively. Persistent-loss tests examine session handling after retries are exhausted. UAV recovery time is measured from the start of the rekeying transaction until the UAV confirms completion; unsuccessful updates are excluded from successful recovery times.

Table 8 shows that normal rekeying averaged 24.21 ms. All first-loss R1–R4 cases recovered in 227.57–243.53 ms; two consecutive losses of R2 or R4 required approximately 425 ms. Duplicate, reordering, and continuous-update tests behaved as expected. Under persistent R2 loss, the old session was retained, whereas persistent R4 loss terminated SID, leaving only 415 of 636 control commands received. QUAD therefore handles bounded link

faults, but cannot guarantee uninterrupted communication when key acknowledgments are persistently lost.

Table 8. Key-update and failure-recovery results.

Test Scenario	Rekey Outcome	UAV Recovery Time (ms)	UAV→GCS Telemetry Received/Sent	GCS→UAV Commands Received/Sent
Normal rekeying	5/5 successful	24.21 ± 17.07	1225/1225	1225/1225
First loss of R1–R4	20/20 successful	227.57–243.53	1225/1225 in each case	1225/1225 in each case
Two consecutive R2/R4 losses	10/10 successful	424.63–427.34	1225/1225 in each case	1225/1225 in each case
Duplicate R1–R4	20/20 successful	24.49–33.32	1225/1225 in each case	1225/1225 in each case
R2 reordering/stale R2	10/10 successful	225.99–258.09	1225/1225 in each case	1225/1225 in each case
Persistent R2 loss	0/5; old session retained	—	1225/1225	1225/1225
Persistent R4 loss	0/5; SID terminated	—	415/415	415/636
10 Consecutive Rekeying Operations (5 Runs)	50/50 Successful	20.69 ± 2.65	2975/2975	2975/2975
20 Consecutive Rekeying Operations (5 Runs)	100/100 Successful	21.16 ± 1.54	6225/6225	6225/6225

Full QUAD was compared with a no-old mode in which old-epoch data reception is disabled. Both use identical R1–R4 messages, retransmissions, key derivation, and state-transition rules; they differ only in old-epoch reception during the transition. Normal network conditions, first R4 loss, random packet losses of 5%, 10%, and 20%, and a fixed delay of 100 ms were tested in five runs per condition. Bidirectional reception rates are shown in Figure 13.

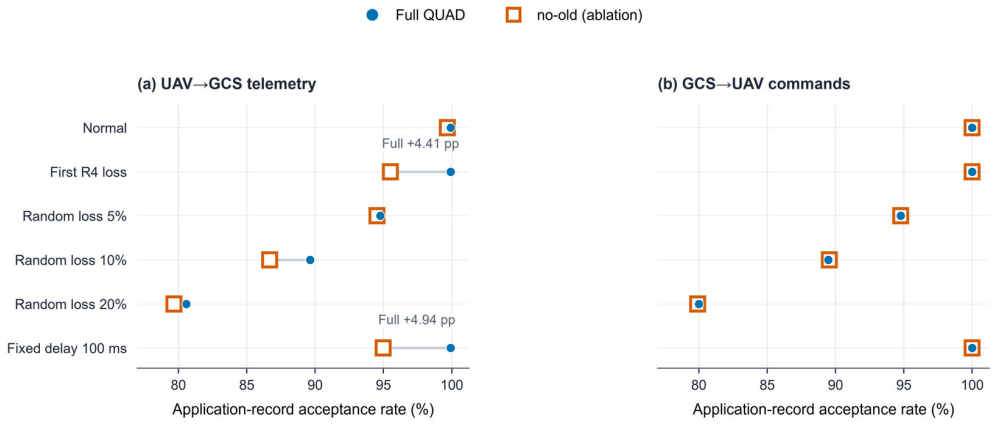


Figure 13. Old-epoch reception ablation results.

Figure 13 shows telemetry reception rates of 99.918% and 95.510% for QUAD and no-old, respectively, under the first R4 loss, and 99.915% and 94.979% under a fixed 100 ms delay. The old-epoch reception window reduces legitimate telemetry losses caused by key-state desynchronization, but improvements vary under random packet loss, and control-command reception does not exhibit an equally large gain.

A Single-key hard-switching baseline was constructed to assess the impact of key-switching policy. It retained ML-KEM and the R1–R4 procedure, but the UAV switched

immediately after sending R3 and the GCS after sending R4, with neither accepting old-epoch records thereafter. Unlike the no-old ablation, this baseline changes both switching times and old-epoch reception. Normal operation, first R3 or R4 loss, 5% and 20% random loss, and fixed 100 ms delay were each repeated five times (Figure 14).

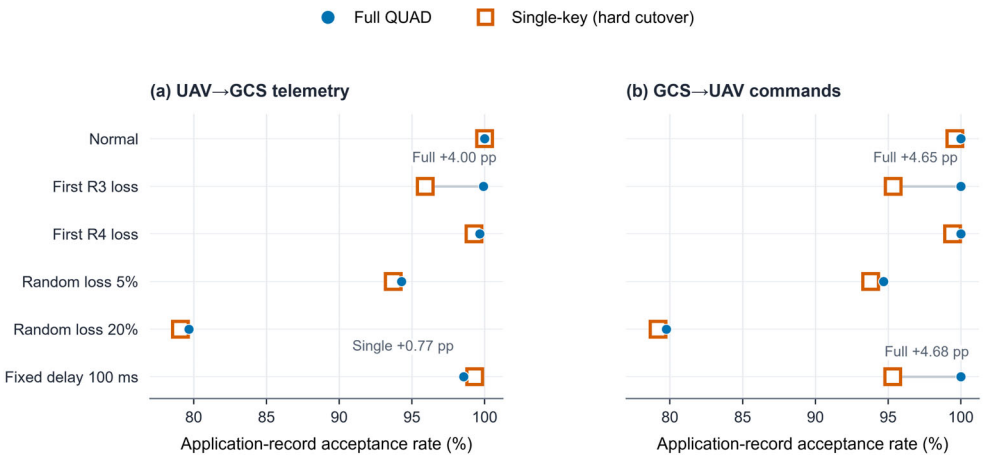


Figure 14. QUAD versus Single-key comparison.

After the first R3 loss, telemetry and control-command reception with QUAD reached 99.918% and 100%, compared with 95.918% and 95.347% for Single-key. Under a 100 ms delay, QUAD achieved 100% command reception versus 95.319% for Single-key, although its telemetry reception was slightly lower. These results indicate that phased switching and bounded old/new-epoch reception can reduce desynchronization and improve bidirectional delivery under selected acknowledgment-loss conditions.

Communication recovery was further compared under controlled bidirectional UDP outages lasting 800 and 1600 ms. Each scheme was tested twice per condition, with three outages and ten planned key updates per run. Metrics included packet reception ratios, mean time from outage end to the first accepted authenticated record in each direction, and counts of successful, failed, and unattempted updates. A recovery count of 6/6 denotes restored bidirectional traffic after all six outages; 20 denotes the total planned updates across both runs. Table 9 summarizes the results.

Table 9. QUAD and Single-key recovery under link outages.

Outage	Scheme	Telemetry (%)	Commands (%)	Both Recovered	Telemetry Delay (ms)	Command Delay (ms)	Rekeys (S/F/N)
800 ms	Full QUAD	92.52	92.66	6/6	22.95	21.27	20/0/0
800 ms	Single-key	82.30	81.29	6/6	32.63	31.06	20/0/0
1600 ms	Full QUAD	84.68	85.04	6/6	37.67	30.64	20/0/0
1600 ms	Single-key	64.76	74.17	6/6	41.33	45.76	20/0/0

For both 800 and 1600 ms outages, QUAD and Single-key restored bidirectional communication in all six windows, and all 20 planned rekeying transactions completed successfully in each reported group. Relative to Single-key, QUAD increased telemetry reception by 10.22 and 19.92 percentage points and command reception by 11.37 and 10.87 percentage points, respectively, while yielding lower mean recovery delays for both traffic directions. Under these tested conditions, phased key switching and bounded overlap reduced the loss of valid application records after interruptions.

7. Discussion

QUAD was evaluated for post-quantum key establishment, invalid-request handling, and rekeying on constrained UAV–GCS links. In the specified ProVerif models, the tested confidentiality and authentication properties hold. Disclosure of one long-term secret after M6 did not expose historical test data, whereas simultaneous disclosure of both long-term materials and future impersonation following LTSK compromise produced the expected counterexamples.

QUAD's mean full-handshake latency was 51.077 ms, above those of the selected comparison schemes. Under invalid-Cookie traffic, however, its process CPU time was approximately 16.4% lower than that of HMAC-only checking. Valid-Cookie requests with incorrect MACs could incur additional computation. Thus, pre-authentication reduces the cost of particular invalid requests but does not eliminate DoS risks.

For rekeying, phased switching and bounded old/new-epoch reception mitigated the effects of lost acknowledgments and inconsistent key states. Ablation and Single-key comparisons showed fewer lost legitimate records and higher bidirectional reception in selected first-R3/R4-loss conditions. Persistent confirmation loss could nevertheless lead to session termination.

The experiments were conducted primarily in a single-UAV setup using a Raspberry Pi and PC. Interference during actual flights, multi-UAV concurrency, and energy costs remain insufficiently evaluated. Future work will examine long-term operation over real wireless links, improve recovery under persistent loss, and assess QUAD on additional constrained platforms.

The main terminology used in this paper is summarized in Appendix A.

Author Contributions: Conceptualization, L.L. and J.T.; methodology, J.T. and L.L.; software, J.T.; validation, J.T., J.L. and R.Z.; formal analysis, J.T.; investigation, J.T.; resources, L.L.; data curation, J.T. and R.Z.; writing—original draft preparation, J.T.; writing—review and editing, L.L., J.T., J.L., R.Z. and H.W.; visualization, J.T. and J.L.; supervision, L.L. and H.W.; project administration, L.L.; funding acquisition, L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China, grant number 2024YFC3014400; the Fundamental Research Funds for the Central Universities, grant Number 25CAFUC09013; the Sichuan Civil Aviation Flight Technology and Flight Safety Engineering Technology Research Center Project, grant numbers GY2024-25D, GY2025-51E; and the Innovation Training Program of Civil Aviation Flight University of China, grant number X202510624433.

Data Availability Statement: The data supporting the findings of this study are not publicly available due to project confidentiality and security restrictions.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A. Protocol Symbols and Parameters

Table A1. System entities.

Symbol	Meaning	Length or Location
U_i,G	The i-th UAV and the ground control station (GCS)	Communication entities; not protocol secrets
PID_i	Pseudonymous identifier of UAV_i	Pre-provisioned at both parties; transmitted in M3; encoded length is implementation-defined
ID_G	GCS identity context	Pre-provisioned at both parties; included in the MAC input; not transmitted as a separate M3 field

Table A1. *Cont.*

Symbol	Meaning	Length or Location
$LTSK_i$	Long-term shared key assigned exclusively to UAV_i	256 bits; stored only by UAV_i and the GCS
MK	GCS Cookie master key	256 bits; generated and stored by the GCS; not transmitted
pk_G	GCS ML-KEM-768 public key	Pre-provisioned at the UAV and held by the GCS; not transmitted during this handshake
sk_G	GCS ML-KEM-768 private key	Stored only by the GCS; not transmitted
IP_{src}	Source network address observed by the GCS	Obtained from the GCS network stack; not included as an M3 field
$t_{current}$	Current local time at the GCS	Used to compute the Cookie time slot; not transmitted

Table A2. Cookie and M1–M3 initial authentication parameters.

Symbol	Meaning	Length/Transmission
Δ_{cookie}	Duration of the time slot used to generate a Cookie	Implementation-configured parameter; not transmitted
T_{slot}	Time slot associated with the Cookie	Integer; transmitted in M2 and M3
N_G	Handshake random value generated by the GCS	128 bits; transmitted in M2 and M3
N_U	Handshake random value generated by the UAV	128 bits; transmitted in M3
Cookie	$HMAC_{MK}(E(L_C, IP_{src}, T_{slot}, N_G))$	HMAC-SHA-256 output; transmitted in M2 and M3
ct, ss	ML-KEM-768 encapsulation ciphertext and shared secret for the initial handshake	ct is transmitted in M3; ss is retained locally by both parties
MAC_U	HMAC over the authentication context computed by the UAV using $LTSK_i$	256 bits; transmitted in M3
HID	Handshake transaction identifier derived by the GCS from an authenticated M3	256 bits; used locally by the GCS for duplicate detection; not transmitted

Table A3. Handshake transcript and key derivation parameters.

Symbol	Meaning	Length/Property
TH_3	Transcript hash of the canonically encoded M1–M3 messages using $E(\cdot)$	256 bits; public digest
TH_4	Transcript hash derived from TH_3 and the encoding of M4	256 bits; public digest
TH_5	Transcript hash derived from TH_4 and the encoding of M5	256 bits; public digest
K_{sym}	Symmetric handshake material derived from $LTSK_i$ and TH_3 using HMAC	256 bits; secret
PRK	Initial pseudorandom key obtained from HKDF-Extract (K_{sym}, ss)	256 bits; secret

Table A3. *Cont.*

Symbol	Meaning	Length/Property
SID	Session identifier derived from TH_3 and a session-ID label	256 bits; transmitted in M4/M5 and record headers
$K_{\mathrm{Fin},G}$	GCS Finished confirmation key derived from PRK	256 bits; secret; not transmitted
$\mathrm{Finished}_G$	HMAC over TH_3 computed by the GCS using $K_{\mathrm{Fin},G}$	256 bits; transmitted in M4
$K_{\mathrm{Fin},U}$	UAV finished confirmation key derived from PRK	256 bits; secret; not transmitted
$\mathrm{Finished}_U$	HMAC over TH_4 computed by the UAV using $K_{\mathrm{Fin},U}$	256 bits; transmitted in M5
$\mathrm{ES}_0, \mathrm{ES}_e$	Initial epoch secret and the secret for epoch e	256 bits; secret; not transmitted
K^e_{U2G}	UAV-to-GCS data-encryption key for epoch e	256 bits; secret; not transmitted
K^e_{G2U}	GCS-to-UAV data-encryption key for epoch e	256 bits; secret; not transmitted

Table A4. Encrypted record, encoding, and replay protection parameters.

Symbol	Meaning	Length/Rule
$\mathrm{H}(\cdot)$	SHA-256 hash function	256-bit output
$\mathrm{E}(\cdot)$	Canonical encoding with a specified field order and explicit length boundaries	Both parties must use byte-for-byte identical inputs
I2OSP(x,n)	Integer-to-octet-string conversion using an n-byte big-endian representation	Epoch occupies 4 bytes and Seq 8 bytes in the nonce
Header	{SID, Direction, Epoch, Seq, Type, Length}	Transmitted in plaintext and authenticated as AAD
AAD	E(Header)	AES-GCM additional authenticated data
Direction	Communication direction of a record	U2G or G2U
Epoch,Seq	Epoch number and message sequence number within that direction and epoch	32 bits and 64 bits, respectively
Nonce	$\mathrm{I2OSP}(\mathrm{Epoch}, 4) \parallel \mathrm{I2OSP}(\mathrm{Seq}, 8)$	96 bits; must not repeat under the same data-encryption key
Type,Length	Record type and length of the encrypted payload	Fixed enumeration/fixed-width encoded integer
Ciphertext,Tag	AES-256-GCM ciphertext and authentication tag	Tag is 128 bits (16 bytes)
$\mathrm{Seq}_{\mathrm{tx}}^{(\mathrm{dir},e)}$	Transmit sequence-number state for the specified direction and epoch	One counter shared by all record types
$\mathrm{Seq}_{\mathrm{max}}^{(\mathrm{dir},e)}$	Highest valid sequence number accepted by the receiver	Maintained separately for each (SID, Direction, Epoch)
$W, \mathrm{Bitmap}^{(\mathrm{dir},e)}$	Replay-window width and corresponding sliding bitmap	W is an implementation parameter; updated only after successful tag verification

Table A5. Rekeying transaction and new key parameters.

Symbol	Meaning	Length/Property
e, e'	Current and target epochs, where e′ = e + 1	32 bits; public
RID	Identifier of the current rekeying transaction	128 bits; transmitted in the R1–R4 payloads
ct′, ss′	New ML-KEM ciphertext and shared secret for rekeying	ct′ is in the encrypted R1 payload; ss′ is retained locally
RH	Rekeying-context hash binding SID, e, e′, RID, and ct′	256 bits; public digest
RPRK′	Rekeying pseudorandom key obtained from HKDF-Extract (ES_e, ss′)	256 bits; secret
$ES_{e'}$	New epoch secret derived from RPRK′ and the rekeying context	256 bits; secret
$K_{U2G}^{e'}$	New UAV-to-GCS data-encryption key for epoch e′	256 bits; secret
$K_{G2U}^{e'}$	New GCS-to-UAV data-encryption key for epoch e′	256 bits; secret
K_{RK}	Rekeying confirmation key derived from $ES_{e'}$	256 bits; secret
KC_G	GCS proof of possession of the new key, computed using K_{RK}	256 bits; transmitted in the encrypted R2 payload

Table A6. Fixed public domain-separation labels.

Symbol	Purpose
L_C	Cookie generation
L_A	UAV identity authentication MAC_U
L_K	Derivation of the symmetric handshake material K_{sym}
L_{SID}	Derivation of session identifier SID
L_{FG}	Derivation of the GCS Finished confirmation key
L_{FU}	Derivation of the UAV Finished confirmation key
L_{ES}	Derivation of the epoch secret
L_{U2G}	Derivation of the UAV-to-GCS data-encryption key
L_{G2U}	Derivation of the GCS-to-UAV data-encryption key
L_R	Rekeying context RH
L_{RK}	Derivation of rekeying confirmation key K_{RK}

Table A7. Local cache and resource-limit parameters.

Symbol	Meaning and Function
$T_{handshake}$	Maximum handshake lifetime; a new handshake must be initiated after expiration
T_{HID}	Retention period for HID duplicate-detection cache entries
T_{RID}	Retention period for processed RID transactions and completion records
RTO	Retransmission timeout interval
N_{retry}	Maximum number of retransmissions allowed for one transaction
T_{grace}	Maximum grace period for bounded overlapping reception of old and new epochs
$N_{pending}^{PID}$	Maximum number of pending transactions allowed for a single PID
$N_{pending}^{global}$	Maximum number of pending transactions that the GCS may store concurrently

References

1. Abdelmaboud, A. The Internet of Drones: Requirements, Taxonomy, Recent Advances, and Challenges of Research Trends. *Sensors* **2021**, *21*, 5718. [CrossRef] [PubMed]
2. Ceviz, O.; Sen, S.; Sadioglu, P. A Survey of Security in UAVs and FANETs: Issues, Threats, Analysis of Attacks, and Solutions. *IEEE Commun. Surv. Tutor.* **2025**, *27*, 3227–3265. [CrossRef]
3. Khan, M.A.; Javaid, S.; Mohsan, S.A.H.; Tanveer, M.; Ullah, I. Future-Proofing Security for UAVs With Post-Quantum Cryptography: A Review. *IEEE Open J. Commun. Soc.* **2024**, *5*, 6849–6871. [CrossRef]
4. Nowakowski, M.; Berger, G.S.; Braun, J.; Mendes, J.A.; Bonzatto Junior, L.; Lima, J. Advance Reconnaissance of UGV Path Planning Using Unmanned Aerial Vehicle to Carry Our Mission in Unknown Environment. In *Proceedings of the Robot 2023: Sixth Iberian Robotics Conference, Coimbra, Portugal, 22–24 November 2023*; Marques, L., Santos, C., Lima, J.L., Tardioli, D., Ferre, M., Eds.; Lecture Notes in Networks and Systems; Springer Nature: Cham, Switzerland, 2024; Volume 978, pp. 50–61, ISBN 978-3-031-59166-2.
5. Espinosa Pedroza, A.D.; Vazquez Trejo, J.A.; Ortiz-Torres, G.; Alonso-Bastida, A.; Borja-Jaimes, V.; Rios-Enriquez, C.; Peregrino, C.A.; Adam-Medina, M. Leader-following Consensus Control Robust to External Disturbances for LPV Multi-agent Systems. *Asian J. Control* **2026**, *28*, 2165–2178. [CrossRef]
6. Borja-Jaimes, V.; Valdez-Martínez, J.S.; Beltrán-Escobar, M.; Ramírez-Zúñiga, G.; Reyes-Mayer, A.; Calixto-Rodríguez, M. Robust Backstepping-Sliding Control of a Quadrotor UAV with Disturbance Compensation. *Computation* **2026**, *14*, 51. [CrossRef]
7. Li, S.; Guo, H.; Liu, J.; Han, Y.; Song, H. An Ultra-Lightweight PUF and ASCON-Based Authentication and Key Agreement Protocol for UAV-Ground Station and UAV-UAV Communication. *Cybersecurity* **2026**, *9*, 164. [CrossRef]
8. Wang, Z.; Li, J.; Yuan, S.; Zhang, Y. An Efficient Group Key Agreement Scheme for UAV Swarms. *Front. Phys.* **2026**, *14*, 1698126. [CrossRef]
9. Xiao, M.; Wan, H.; Fan, H.; Shao, H.; Li, Z.; Yang, K. Formal Analysis of Ratchet Protocols Based on Logic of Events. *Appl. Sci.* **2025**, *15*, 6964. [CrossRef]
10. Dolev, D.; Yao, A. On the Security of Public Key Protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
11. Blanchet, B. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Found. Trends® Priv. Secur.* **2016**, *1*, 1–135. [CrossRef]
12. Wazid, M.; Das, A.K.; Kumar, N.; Vasilakos, A.V.; Rodrigues, J.J.P.C. Design and Analysis of Secure Lightweight Remote User Authentication and Key Agreement Scheme in Internet of Drones Deployment. *IEEE Internet Things J.* **2019**, *6*, 3572–3584. [CrossRef]
13. Hussain, S.; Chaudhry, S.A.; Alomari, O.A.; Alsharif, M.H.; Khan, M.K.; Kumar, N. Amassing the Security: An ECC-Based Authentication Scheme for Internet of Drones. *IEEE Syst. J.* **2021**, *15*, 4431–4438. [CrossRef]
14. Srinivas, J.; Das, A.K.; Kumar, N.; Rodrigues, J.J.P.C. TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6903–6916. [CrossRef]
15. Jan, S.U.; Qayum, F.; Khan, H.U. Design and Analysis of Lightweight Authentication Protocol for Securing IoD. *IEEE Access* **2021**, *9*, 69287–69306. [CrossRef]
16. Park, Y.; Ryu, D.; Kwon, D.; Park, Y. Provably Secure Mutual Authentication and Key Agreement Scheme Using PUF in Internet of Drones Deployments. *Sensors* **2023**, *23*, 2034. [CrossRef] [PubMed]
17. Bera, B.; Das, A.K.; Garg, S.; Piran, M.J.; Hossain, M.S. Access Control Protocol for Battlefield Surveillance in Drone-Assisted IoT Environment. *IEEE Internet Things J.* **2022**, *9*, 2708–2721. [CrossRef]
18. Shariq, M.; Jamil, N.; Rawat, G.S.; Chaudhry, S.A.; Masud, M.; Das, A.K. Provably Secure and Reliable Privacy-Preserving Authentication Scheme for Drone-to-Drone Communications in Internet of Autonomous Things. *IEEE Trans. Mob. Comput.* **2026**, *25*, 1445–1456. [CrossRef]
19. Illyass, K.; Baig, Z.; Syed, N. Robust and Lightweighted Mutual Authentication Scheme for Drone Swarm Networks. *J. Netw. Comput. Appl.* **2025**, *242*, 104264. [CrossRef]
20. Message Signing (Authentication). Available online: https://mavlink.io/en/guide/message_signing.html (accessed on 15 October 2026).
21. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS—Kyber: A CCA-Secure Module-Lattice-Based KEM. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018*; IEEE: London, UK, 2018; pp. 353–367.
22. National Institute of Standards and Technology (US). *Module-Lattice-Based Key-Encapsulation Mechanism Standard*; National Institute of Standards and Technology (U.S.): Washington, DC, USA, 2024.
23. Tawfeeq, N.H.; Yousif, M.; Al-Shareeda, M.A.; Almaiah, M.A.; Shehab, R. Lightweight and Quantum-Resistant Authentication for the Internet of Drones (IoD) Using Dilithium Signatures. *Int. J. Innov. Res. Sci. Stud.* **2025**, *8*, 2842–2853. [CrossRef]
24. He, L.; Zhao, M.; Wang, X.; Wang, J.; Wang, Z.; Liu, S. A Post-Quantum Authentication and Key Agreement Scheme for Drone Swarms. *Electronics* **2025**, *14*, 3364. [CrossRef]

25. Nair, A.S.; Thampi, S.M.; Jafeel, V. A Post-Quantum Secure PUF Based Cross-Domain Authentication Mechanism for Internet of Drones. *Veh. Commun.* **2024**, *47*, 100780. [CrossRef]
26. Xu, P.; Wu, H.; Tao, X.; Wang, C.; Chen, D.; Nan, G. Anti-Quantum Certificateless Group Authentication for Massive Accessing IoT Devices. *IEEE Internet Things J.* **2024**, *11*, 16561–16577. [CrossRef]
27. Mishra, D.; Singh, M.; Rewal, P.; Pursharthi, K.; Kumar, N.; Barnawi, A.; Rathore, R.S. Quantum-Safe Secure and Authorized Communication Protocol for Internet of Drones. *IEEE Trans. Veh. Technol.* **2023**, *72*, 16499–16507. [CrossRef]
28. Xia, T.; Wang, M.; He, J.; Yang, G.; Fan, L.; Wei, G. A Quantum-Resistant Identity Authentication and Key Agreement Scheme for UAV Networks Based on Kyber Algorithm. *Drones* **2024**, *8*, 359. [CrossRef]
29. Chaudhary, D.; Lee, C.-C. Anonymous Quantum-Safe Secure and Authorized Communication Protocol under Dynamic Identities for Internet of Drones. *Comput. Electr. Eng.* **2024**, *120*, 109774. [CrossRef]
30. Schwabe, P.; Stebila, D.; Wiggers, T. Post-Quantum TLS Without Handshake Signatures. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual, 30 October 2020; ACM: New York, NY, USA, 2020; pp. 1461–1480.
31. Han, S.; Jager, T.; Kiltz, E.; Liu, S.; Pan, J.; Riepel, D.; Schäge, S. Authenticated Key Exchange and Signatures with Tight Security in the Standard Model. In *Proceedings of the Advances in Cryptology—CRYPTO 2021, Virtual, 16–20 August 2021*; Malkin, T., Peikert, C., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2021; Volume 12828, pp. 670–700. ISBN 978-3-030-84258-1.
32. Alwen, J.; Blanchet, B.; Hauck, E.; Kiltz, E.; Lipp, B.; Riepel, D. Analysing the HPKE Standard. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2021, Zagreb, Croatia, 17–21 October 2021*; Canteaut, A., Standaert, F.-X., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2021; Volume 12696, pp. 87–116. ISBN 978-3-030-77869-9.
33. Zhou, B.; Jiang, H.; Zhao, Y. CPA-Secure KEMs Are Also Sufficient for Post-Quantum TLS 1.3. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2024, Kolkata, India, 9–13 December 2024*; Chung, K.-M., Sasaki, Y., Eds.; Lecture Notes in Computer Science; Springer Nature Singapore: Singapore, 2025; Volume 15486, pp. 433–464, ISBN 978-981-96-0890-4.
34. Liu, Y.; Zhou, B.; Jiang, H. CuKEM: A Concise and Unified Hybrid Key Encapsulation Mechanism. In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, Taipei, Taiwan, 19 November 2025; ACM: New York, NY, USA, 2025; pp. 4707–4721.
35. Xu, G.; Yin, X.; Li, X. ER-CGKA: Efficient and Robust Continuous Group Key Agreement Scheme with Post-Compromise Forward Security for IoV. *PLoS ONE* **2024**, *19*, e0307867. [CrossRef] [PubMed]
36. Ko, Y.; Kim, J.; Duguma, D.G.; Astillo, P.V.; You, I.; Pau, G. Drone Secure Communication Protocol for Future Sensitive Applications in Military Zone. *Sensors* **2021**, *21*, 2057. [CrossRef] [PubMed]
37. Kaufman, C.; Hoffman, P.; Nir, Y.; Eronen, P.; Kivinen, T. *Internet Key Exchange Protocol Version 2 (IKEv2)*; RFC Editor: Marina del Rey, CA, USA, 2014.
38. Rescorla, E.; Tschofenig, H.; Modadugu, N. *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3*; RFC Editor: Marina del Rey, CA, USA, 2022.
39. Donenfeld, J.A. WireGuard: Next Generation Kernel Network Tunnel. In *Proceedings of the 2017 Network and Distributed System Security Symposium, San Diego, CA, USA, 26 February–1 March 2017*; Internet Society: San Diego, CA, USA, 2017.
40. Varner, K.; Lipp, B.; Zaeske, W.; Schmidt, L.; Dua, P. Rosenpass; Rosenpass e.V. 2026. Available online: https://rosenpass.eu/whitepaper.pdf (accessed on 15 October 2026).
41. Selander, G.; Preuß Mattsson, J.; Palombini, F. *Ephemeral Diffie-Hellman Over COSE (EDHOC)*; RFC Editor: Marina del Rey, CA, USA, 2024.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.