*electronics*

MDPI

*Article*

# A Nonce-Bound Lightweight Mutual Authentication Protocol for UAV-Assisted Smart Mining Systems

**Hao Li [1,2], Zhenhu Ning [3,*], Muhammad Waqas [4,*] and Hisham Alfuhaid [5,6]**

1 School of Ocean Sciences and Resources, China University of Geosciences (Beijing), Beijing 100083, China; lihaocean@cugb.edu.cn
2 Key Laboratory of Polar Geology and Marine Mineral Resources [China University of Geosciences (Beijing)], Ministry of Education, Beijing 100083, China
3 College of Cyberspace Science and Technology, Beijing University of Technology, Beijing 100083, China
4 School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK
5 Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK; alasmary@kku.edu.sa
6 Department of Computer Science, King Khalid University, Abha 61421, Saudi Arabia
* Correspondence: ningzhenhu@bjut.edu.cn (Z.N.); engr.waqas2079@gmail.com (M.W.)

**Abstract**

Unmanned aerial vehicles (UAVs) support surveying, slope monitoring, inspection, and emergency observation in smart mines, but their wireless access links expose operational data and commands for impersonation, replay, modification, and tracking. This paper presents a lightweight mutual authentication protocol for a three-entity mining architecture consisting of a UAV, an edge gateway, and a mine control center. The protocol uses HMAC-SHA-256, fresh nonces, dynamic pseudonyms, and a dual-slot backend state. A server response is bound to the active gateway challenge, and per-UAV state lookup, replay checking, and updates are performed atomically. These rules reject delayed responses and serialize backend updates. A Real-or-Random analysis considers session-key secrecy under uncompromised endpoints and protected backhaul transport; final-response-loss recovery additionally assumes that backend requests are committed in attempt order. Transition-level tests cover normal completion, final-message loss and recovery, delayed response, replay, and authenticator modification. Performance analysis derives the proposed operation and communication costs and compares them with five published UAV authentication protocols. The proposed UAV performs one nonce generation and five HMAC evaluations; its authentication protocol payload is 1024 bits on the UAV-facing radio link and 2048 bits across all five link transmissions, excluding link/transport headers and protected-backhaul cryptographic overhead.

**Keywords:** UAV authentication; smart mining; HMAC; dynamic pseudonym; edge gateway; desynchronization recovery

Check for updates

Academic Editor: Aryya Gangopadhyay

Received: 9 October 2026
Revised: 5 November 2026
Accepted: 12 November 2026
Published: 18 November 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.

## 1. Introduction

Smart mining systems combine mobile sensing, edge processing, and control-center decision support. UAVs collect high-resolution imagery, point clouds, thermal observations, and inspection records over areas that are costly or unsafe for manual access. Mining-oriented studies report UAV use in surveying, stockpile measurement, pit-wall mapping, slope inspection, tailings monitoring, and rehabilitation surveillance [1–5]. Mine digital-twin and edge-node studies further show that these data are increasingly connected to operational platforms rather than processed as isolated survey products [6]. This connectivity creates an authentication problem. A UAV may upload slope images, road conditions,

or emergency observations and may receive routes, sensing tasks, and return commands. An adversary that impersonates a UAV, replays stale observations, modifies a command, or tracks a persistent identifier can affect operational decisions or expose mine activity [7–9]. Mine sites also have intermittent wireless coverage due to benches, terrain, machinery, dust, and electromagnetic interference [10,11]. Therefore, a useful protocol must handle final-message loss without leaving the UAV and backend permanently desynchronized. Published Internet of Drones (IoD) protocols use temporal credentials, authenticated encryption, physical unclonable functions (PUFs), fuzzy extractors, and hash-based constructions [12–17]. Their architectures mainly authenticate a mobile user to a drone through a ground server or authenticate a UAV directly to a ground station. In contrast, the target mine separates local radio access at an edge gateway from identity and state management at a mine control center. This separation requires explicit transaction binding across the wireless and protected backhaul links [18–20].

Compromised UAV information can affect specific mine workflows. Forged slope or tailings observations may distort hazard screening, stale inspection imagery may conceal a recent change, and modified routes or return commands may divert an inspection [21–23]. Substituted stockpile measurements may also bias inventory estimates. These are plausible operational consequences rather than incidents demonstrated here. Authentication establishes the registered source and session integrity; it does not certify sensor accuracy, route safety, or the correctness of downstream decisions [24,25].

Related security designs also address authentication and protected data exchange in other connected-device settings. Tan et al. investigate ASCON-based group authentication and key agreement with non-linkability and integrity assurance for industrial Internet of Things (IIoT) systems [26]. Sourav et al. propose lattice-based certificateless signcryption using blockchain for smart healthcare systems [27]. These studies broaden the context to group authentication and blockchain-supported healthcare data protection. The present work focuses on per-UAV mutual authentication and recovery from interrupted credential updates in a trusted gateway–control-center architecture.

The main contributions are as follows:

- A three-entity smart-mining authentication model is defined with an exposed UAV–gateway link, a protected gateway–control-center backhaul, and explicit placement of long-term state.
- A nonce-bound protocol is constructed from HMAC-SHA-256 with domain-separated inputs. The final server token carries the active gateway nonce, and the control center uses atomic state transitions to serialize credential updates; the loss-recovery claim is conditional on backend request ordering.
- A dual-slot update rule is provided for recovery after loss of the final response. Its behavior is checked through executable transition tests for normal completion and four adverse message-delivery cases.
- An operation-level performance analysis derives the UAV, gateway, and control-center computation costs and the radio and all-link communication overheads; the resulting values are compared with five published protocols.

The remainder of the paper is organized as follows. Section 2 reviews the application and protocol context. Section 3 defines the system and threat models. Section 4 states the primitives and invariants. Section 5 specifies the protocol. Sections 6 and 7 analyze security and performance, respectively. Section 8 concludes the paper.

## 2. Related Work

### *2.1. UAV-Assisted Smart Mining*

Mining UAV research has primarily addressed sensing accuracy and operational efficiency. Photogrammetry and LiDAR support three-dimensional mapping and deformation monitoring [4,28]; UAV imagery supports pit-wall geological mapping [29]; and repeated flights support mine-slope and landslide observation [30,31]. UAV-assisted mine-closure surveillance and image-based mine-area classification have also been investigated [32–34]. These studies establish the operational scenario, but authentication is not their main technical contribution. When UAV observations feed a mine platform, provenance becomes part of the control workflow. Digital-twin research connects sensing to simulation and decision support [35], while smart-mine edge deployment targets low-latency access to backend services [6]. The control center must therefore authenticate a registered UAV before accepting its data or issuing protected task traffic.

### *2.2. Published UAV Authentication Protocols*

IoD surveys identify eavesdropping, replay, impersonation, message modification, privacy leakage, physical capture, and denial-of-service as recurring threats [12,13]. TCALAS authenticates a user, a ground station server, and a drone using temporal credentials and lightweight operations [14,36]. LAKE-IoD uses SHA-256, AEGIS, and a fuzzy extractor in a mobile-user, management-server, and drone architecture [15,37]. SecAuthUAV uses PUF challenge–response pairs for UAV–ground-station authentication and extends the established channels to UAV–UAV communication [16,38]. SLAP-IoD combines PUFs and a fuzzy extractor in a smart-city IoD access protocol [39–41]. PSLAPS-IoD uses hash, fuzzy extractor, and random operations in a drone, ground server, mobile device, and trusted server setting [17,42]. These protocols demonstrate several viable primitive choices, but their state and message paths cannot be copied directly into the target mine [43]. The proposed method narrows the participants to a registered UAV and a mine control center, with the gateway providing trusted challenge generation, pending-transaction management, and forwarding. It also treats final-response loss and delayed delivery as protocol-state events rather than only as channel errors.

The specific gap is the combined treatment of response-to-transaction binding, atomic updates of a shared backend record, and recovery after loss of the final response in this separated edge/backend architecture. Previous-state retention alone is not claimed as new: SecAuthUAV already retains previous and new PUF challenge–response pairs. The contribution here is the explicit interaction of the transcript, pending state, and dual-slot rules under the stated assumptions, rather than a new primitive or a claim that the cited protocols are insecure.

## 3. System Model and Threat Model

### *3.1. System Model*

The system contains the following entities, as shown in Figure 1.

- **UAV** $U_i$**:** a mobile sensing node used for mine surveying, inspection, environmental monitoring, and emergency observation. It can generate cryptographically secure nonces and evaluate HMAC-SHA-256.
- **Edge gateway** *EG***:** a trusted ground station or edge access point near the operating area. It generates the challenge $N_E$, creates and expires the pending association context, binds a response to that context, and forwards messages. It does not verify UAV HMACs or derive the UAV session key; credential verification belongs to *MCC*. It stores no UAV long-term secret.

- **Mine control center** *MCC*: a trusted backend that stores UAV registration records, validates authentication requests, maintains two credential slots, and derives session keys.

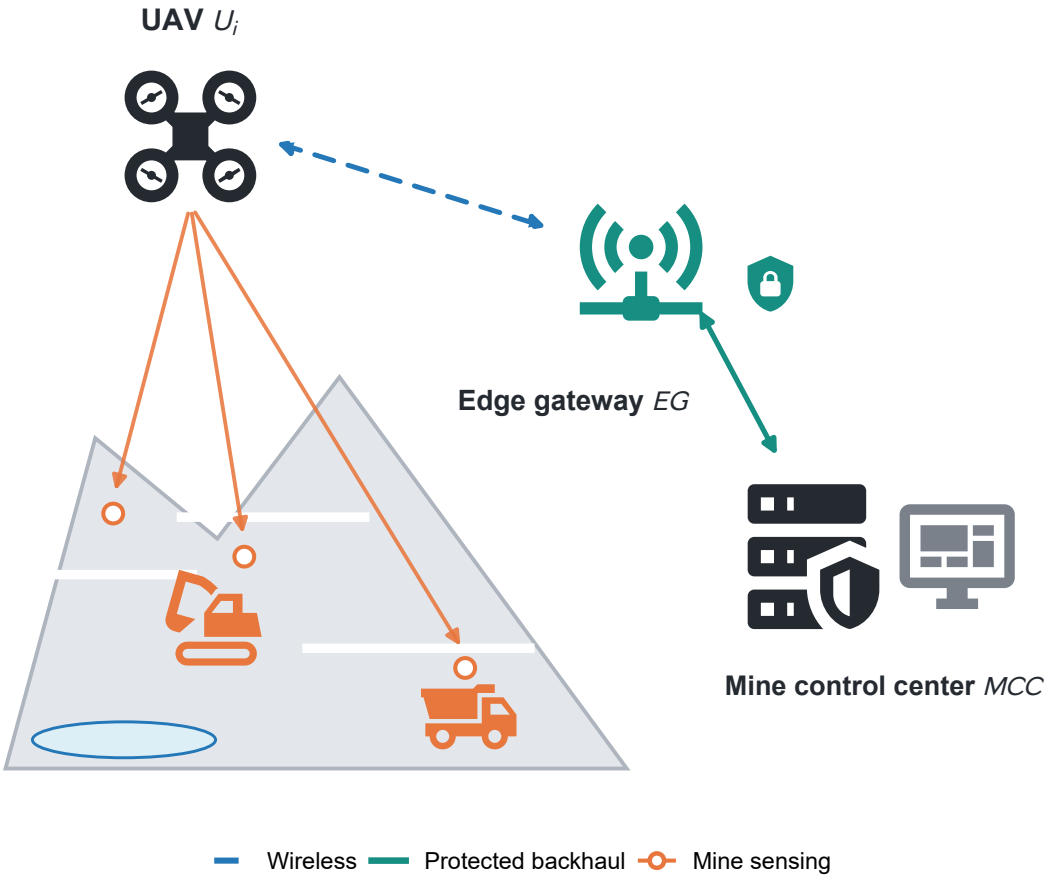



**Figure 1.** System model of the proposed protocol.

After authentication, $U_i$ and *MCC* hold the same session key for protecting task data with an appropriate symmetric authenticated-encryption scheme. The data protection mechanism is outside the authentication transcript. The permanent identity is not transmitted during online authentication.

### *3.2. Threat and Trust Assumptions*

The adversary controls the wireless channel between $U_i$ and *EG*. It can eavesdrop, insert, replay, reorder, modify, delay, and drop wireless messages and can attempt to track repeated identifiers [44]. It can also cause the final response to be lost. This is the standard active-channel capability used in the IoD literature [12,14,15]. The adversary is computationally bounded, and HMAC-SHA-256 is assumed to be a secure pseudorandom function under the secret keys. Forgery and random-collision probabilities are accounted for in Section 6. Control-center compromise is excluded. The *EG*–*MCC* backhaul provides peer authentication, integrity, confidentiality, and replay protection. The gateway follows the protocol and protects its pending-transaction context. Figure 2 separates these domains. Physical extraction of the current UAV secret, gateway compromise, control-center compromise, and denial of radio availability are outside the main guarantee. In particular, no authentication protocol can prevent a jammer from dropping every message. These exclusions are revisited in Section 6. The out-of-band enrollment channel used in Section 5 is assumed to be authentic and confidential; it is realized by factory provisioning or wired on-site injection before the UAV enters online service, and is not exposed to the wireless

adversary. Compromise of this channel is equivalent to long-term key compromise and is out of scope of the online protocol. The loss-recovery claim additionally assumes that an earlier forwarded request cannot commit after a newer attempt for the same UAV has committed. This ordering condition is separate from database atomicity and is discussed in Section 5.4; arbitrary reordering of queued backhaul requests is not covered by that recovery claim. The objectives are request-origin authentication at *MCC*, authentication of *MCC* at the UAV, agreement and secrecy of fresh session keys, replay rejection, and conditional recovery after final-response loss. The UAV authenticates the control center rather than an individual gateway. Backend acceptance does not establish that the UAV received the final response or installed the session key.

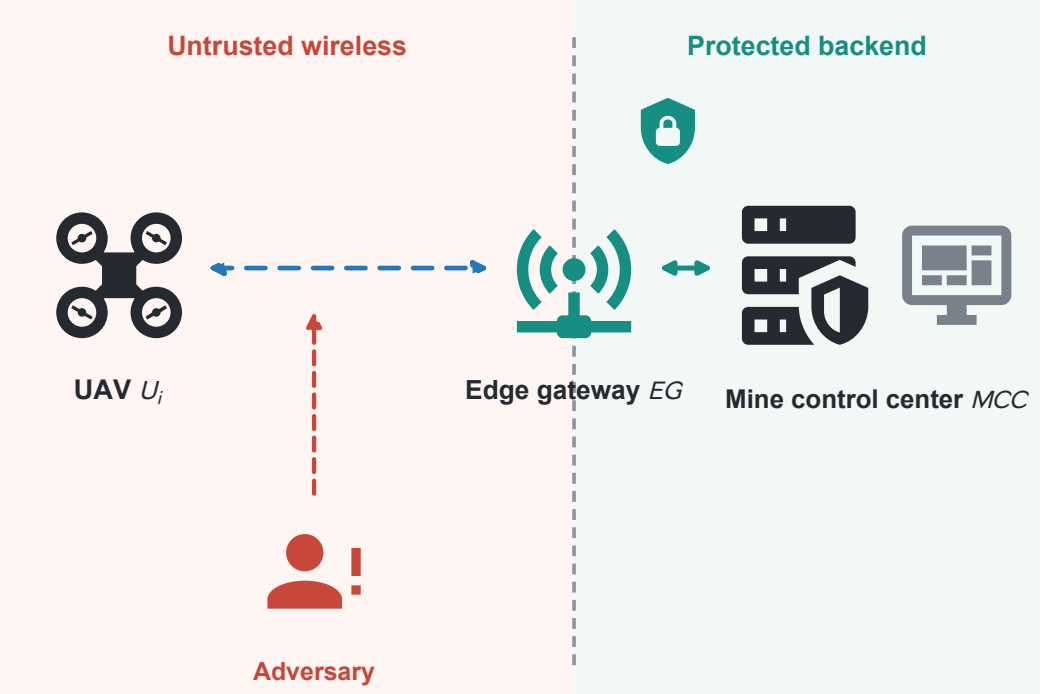



**Figure 2.** Threat model of the proposed protocol.

## 4. Preliminaries and Design Rules

### *4.1. Cryptographic Primitives*

Let $\mathbf{x} = (x_1, \ldots, x_n)$ and define

$$F_K(\tau, \mathbf{x}) = \text{HMAC-SHA-256}\big(K, \text{enc}(\tau, \mathbf{x})\big). \tag{1}$$

Here, $\tau$ is a distinct protocol tag and enc is a canonical, length-delimited encoding. HMAC is used as a keyed message-authentication construction rather than placing a secret directly inside an unkeyed hash [45]. The tags `U`, `SK`, `S`, `PID`, and `KEY` provide domain separation. All comparisons of authentication tokens are constant-time. Calls written as $F_K(\tau, x_1, \ldots, x_n)$ use $\mathbf{x} = (x_1, \ldots, x_n)$ in this definition.

For reproducibility, the byte encoding is fixed as follows. The tags `U`, `SK`, `S`, `PID`, and `KEY` are the single octets `0x01`, `0x02`, `0x03`, `0x04`, and `0x05`, respectively; they are not ASCII strings. Let $\ell_{16}(x)$ encode the byte length of $x$ as an unsigned two-octet big-endian integer. Then

$$\text{enc}(\tau, x_1, \ldots, x_n) = \text{tag}(\tau) \,\|\, \ell_{16}(x_1) \| x_1 \| \cdots \| \ell_{16}(x_n) \| x_n.$$

Every invocation in this protocol has exactly three fields in the order $(PID_i, N_E, N_U)$. Each field is its raw 16-octet value, including leading zero octets, and each length prefix is

therefore `0x0010`. No hexadecimal-text conversion, separator, padding, or terminator is included; the encoded HMAC message is exactly 55 octets. The HMAC key is always the raw 16-octet $K_i$, including for tag `S`: the session key $SK_i$ is an output of the construction and is never itself used as an HMAC key, so the public server authenticator cannot be used to test a candidate session key. Implementations reject an unknown tag, a nonconforming field length or count, or trailing octets. These tags and length prefixes are constructed locally for HMAC input; they are not additional transmitted fields. Online messages use the fixed-width raw fields in the order shown in Section 5.

The function $\mathrm{Trunc}_{128}(x)$ returns the first 128 bits of $x$. Pseudonyms, nonces, and evolving UAV keys are 128 bits; HMAC outputs and session keys are 256 bits. Nonces are generated by a cryptographically secure random-number generator and are never intentionally reused.

Table 1 summarizes the notation.

**Table 1.** Protocol notation.

| Symbol | Description |
|---|---|
| $U_i, EG, MCC$ | UAV, edge gateway, and mine control center |
| $ID_i$ | Permanent identity of $U_i$ |
| $PID_i$ | Current 128-bit dynamic pseudonym |
| $K_i$ | Current 128-bit evolving secret |
| $N_U, N_E$ | UAV and gateway nonces |
| $Auth_U$ | UAV-to-control-center authenticator |
| $Auth_S$ | Control-center-to-UAV authenticator |
| $SK_i$ | 256-bit session key |
| $S_i^{cur}, S_i^{pre}$ | Current and previous backend slots |
| $F_K$ | Domain-separated HMAC-SHA-256 |

### 4.2. State-Machine Invariants

The protocol relies on the four implementation invariants in Table 2. They are part of the protocol definition, not optional deployment advice.

**Table 2.** Protocol-state invariants.

| Rule | Required Behavior | Security Purpose |
|---|---|---|
| I1 | $EG$ and $U_i$ keep at most one active transaction per radio association/UAV. | Prevents an old response from being interpreted under a concurrent transcript. |
| I2 | $U_i$ accepts $M_4$ only when its explicit $N_E$ equals the active pending challenge. | Binds server authentication and credential update to one transaction. |
| I3 | $MCC$ performs lookup, token verification, replay insertion, and dual-slot update in one per-UAV atomic transaction. | Serializes backend read-modify-write operations; it does not impose attempt order on separately queued requests. |
| I4 | Timeout, malformed input, nonce mismatch, or failed token verification causes no persistent UAV state change. | Preserves a recoverable old state under loss, delay, or modification. |

For each UAV, $MCC$ stores

$$S_i^{cur} = (PID_i^{cur}, K_i^{cur}), \qquad S_i^{pre} = (PID_i^{pre}, K_i^{pre}). \tag{2}$$

The previous slot is empty at registration and later contains the last accepted state. It is used only when a UAV did not receive the final response and therefore retries with its old state.

More precisely, after a commit the previous slot holds the exact credential that authenticated that request, even if the request selected the previous slot. Acceptance through

it requires a matching pseudonym, a valid full-transcript HMAC under the retained key, and a tuple absent from the replay cache. The backend cannot observe final-response loss directly; these checks authorize a recovery attempt without proving that loss occurred. Slot rechecking and replay insertion use one lock on the stable UAV record across all gateways.

## 5. Proposed Protocol

### *5.1. Protocol Overview*

The protocol has registration, online mutual authentication, and state-update phases. Figure 3 shows four logical messages. Because $M_4$ traverses both the protected backhaul and the wireless link, the complete architecture performs five link transmissions.

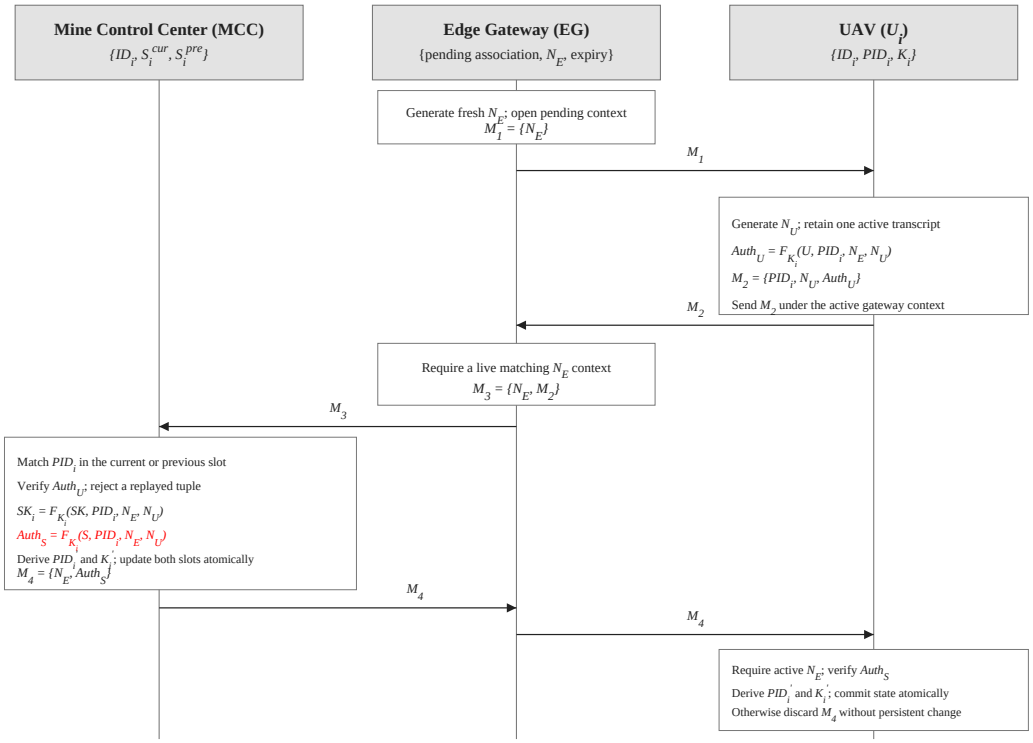



**Figure 3.** Authentication flow of the proposed protocol.

### *5.2. Registration*

Over a secure initialization channel, *MCC* chooses a unique permanent identity $ID_i$, a random pseudonym $PID_i^0$, and a random secret $K_i^0$. The UAV stores $(ID_i, PID_i^0, K_i^0)$. The backend stores

$$\{ID_i, S_i^{cur} = (PID_i^0, K_i^0), S_i^{pre} = \perp\}. \tag{3}$$

The gateway has a site credential for its protected connection to *MCC*, but it receives no UAV secret. This channel is an out-of-band, pre-deployment path (e.g., factory flashing or a wired on-site USB injection) over which *MCC* and the UAV establish $(ID_i, PID_i^0, K_i^0)$ before the UAV is deployed on the mine network. It provides authenticity and confidentiality by physical isolation; it is not a running wireless channel and is not within the adversary's reachable model in Section 3. If this enrollment channel were eavesdropped or tampered with, the adversary would obtain or substitute a valid $(PID_i^0, K_i^0)$ pair and could impersonate that UAV indefinitely until re-enrollment; the online protocol provides no defense against such a supply-chain compromise, by design.

*5.3. Online Authentication*

**Step 1: Gateway challenge.** The gateway generates a fresh $N_E$, opens the pending context $(a, N_E, t_{exp})$ for radio association $a$, and sends

$$M_1 = \{N_E\}. \tag{4}$$

An expired context is deleted, and a new transaction replaces no still-active transaction for the same association.

**Step 2: UAV request.** The UAV generates $N_U$, stores the active transcript $(PID_i, K_i, N_E, N_U)$ in volatile memory, and computes

$$Auth_U = F_{K_i}(\mathtt{U}, PID_i, N_E, N_U). \tag{5}$$

It sends

$$M_2 = \{PID_i, N_U, Auth_U\} \tag{6}$$

to $EG$. The gateway associates $M_2$ with its single pending $N_E$ and forwards

$$M_3 = \{N_E, M_2\} \tag{7}$$

over the protected backhaul. If no live context exists, $M_2$ is discarded.

**Step 3: Atomic backend verification.** The control center begins a per-UAV transaction and finds $PID_i$ in either $S_i^{cur}$ or $S_i^{pre}$. Let the matched state be $(PID_i, K_i)$. It computes

$$Auth_U^* = F_{K_i}(\mathtt{U}, PID_i, N_E, N_U) \tag{8}$$

and rejects unless the constant-time comparison succeeds. A previously accepted tuple $(PID_i, N_E, N_U)$ is also rejected. Only after successful authentication is that tuple inserted into the replay cache. The replay cache is a per-UAV hash table keyed by $(PID_i, N_E, N_U)$. It retains a bounded LRU window of recently accepted tuples (e.g., $W = 16$ entries per UAV) and evicts the oldest entry under memory pressure. An entry is also removed when its $PID_i$ has been absent from both $S_i^{cur}$ and $S_i^{pre}$ and the protected-backhaul replay window has expired. After an entry is evicted, a replayed old $M_2$ cannot succeed: its $N_E$ does not match the gateway's current active challenge (invariant I2), and its $PID_i$ no longer appears in either backend slot, so HMAC verification fails regardless of cache state.

The control center then derives

$$SK_i = F_{K_i}(\mathtt{SK}, PID_i, N_E, N_U), \tag{9}$$

$$Auth_S = F_{K_i}(\mathtt{S}, PID_i, N_E, N_U), \tag{10}$$

$$PID_i' = \mathrm{Trunc}_{128}\big(F_{K_i}(\mathtt{PID}, PID_i, N_E, N_U)\big), \tag{11}$$

$$K_i' = \mathrm{Trunc}_{128}\big(F_{K_i}(\mathtt{KEY}, PID_i, N_E, N_U)\big). \tag{12}$$

The server authenticator uses the evolving secret $K_i$ under the `S` tag. It is separate from the `SK` output, so the public token does not directly verify a candidate application session key. All four outputs above bind the same authenticated transcript.

Within the same transaction, it applies

$$S_i^{pre} \leftarrow (PID_i, K_i), \qquad S_i^{cur} \leftarrow (PID_i', K_i'). \tag{13}$$

The atomic commit succeeds only if the matched slot has not changed since lookup. A unique database index is maintained across all active pseudonym slots. If $PID_i'$ has the negligible event of colliding with an active pseudonym, the transaction is aborted before any state or replay-cache change; a retry under fresh nonces produces another candidate.

Replay entries may be removed after their pseudonym is absent from both slots and the protected-backhaul replay window has expired. The response

$$M_4 = \{N_E, Auth_S\} \tag{14}$$

is then sent to $EG$ and relayed to $U_i$.

**Step 4: UAV acceptance and update.** The UAV first checks that a transaction is active and that the received $N_E$ equals its pending $N_E$. It verifies $Auth_S$ under the pending $K_i$ and transcript, derives $SK_i$ and $(PID_i', K_i')$, and atomically replaces its persistent pair only after all checks pass. It then deletes the pending transcript. A malformed, expired, modified, or nonce-mismatched $M_4$ is discarded without changing persistent state. There is no explicit fifth acknowledgement from the UAV. The first protected application message may provide implicit key confirmation to $MCC$ when required. Consequently, the authentication protocol itself does not claim explicit final key confirmation at the control center. The interface contract between the authentication layer and the application layer is as follows. On the UAV side, the stack returns "authenticated" to the application only after $Auth_S$ is verified and $(PID_i', K_i')$ are committed to persistent storage. On the $MCC$ side, after sending $M_4$ the backend enters an `AUTH_SENT` state and does not accept application data under $SK_i$ until the first protected application message verifies under $SK_i$. If that first application message is delayed or lost, the $MCC$ treats the association as unauthenticated and buffers no business data under the new session key. Consequently, UAV-to-$MCC$ authentication is explicit, while $MCC$-to-UAV key confirmation is implicit and piggybacks on the first application message; the protocol does not claim a third, explicit final-round acknowledgement.

### *5.4. Loss Recovery*

Suppose both parties initially use state $A$. After $MCC$ accepts a request, it stores $A$ as previous and a derived state $B$ as current. If $M_4$ is lost, the UAV remains at $A$. On the next fresh challenge, its valid HMAC under $A$ selects the previous slot. The backend derives a fresh state $C$ from the new nonces and writes $(A, C)$ as its previous/current pair. When the new $M_4$ arrives, the UAV also moves from $A$ to $C$. Thus, under the request-ordering condition in Section 3, a single lost final response does not permanently desynchronize the parties.

Atomicity alone does not establish that ordering condition. For example, let an earlier request and a later retry both use credential $A$. If the retry first commits state $B$ and its response is accepted, a subsequently processed earlier request can still match previous credential $A$ and replace $B$ by $C$, leaving the UAV on $B$. The replay cache does not reject this previously unprocessed tuple. The present counter-free protocol therefore does not claim recovery under arbitrary queued-request reordering. Enforcing attempt order requires an additional deployment mechanism or a protocol extension; it is not established by the five payload transmissions counted here.

We now justify the "at most one prior slot" assumption under repeated $M_4$ loss. Suppose the UAV remains at state $A$ and $MCC$ advances one step to $(S_i^{pre} = A,\ S_i^{cur} = B)$. If $M_4$ is lost again, the UAV retries under fresh nonces still using $A$; the request matches $S_i^{pre}$, and within the atomic transaction $MCC$ rewrites $S_i^{pre} \leftarrow A$ and $S_i^{cur} \leftarrow C$. As long as the UAV has not received any $M_4$, it keeps presenting the same $A$, and $S_i^{pre}$ is pinned to $A$. The previous slot is overwritten only when a request matches $S_i^{cur}$, which requires a valid credential in the current slot; i.e., either a genuine UAV that has already advanced, or an adversary in possession of a non-compromised $K_i$. Hence, under the assumption that $K_i$ is not leaked, the dual-slot rule tolerates an unbounded number of consecutive $M_4$ losses. The hard limits are (i) the UAV's local retransmission timeout, after which it falls back to re-enrollment, and (ii) administrative rollback or database corruption at $MCC$, which are operational rather than protocol concerns.

### *5.5. Transition-Level Validation*

The state-transition implementation was exercised with the cases in Table 3. These tests check protocol control flow; they are not a formal cryptographic proof. The accompanying reference implementation supplies ten reproducible checks, including the listed loss/replay cases, concurrent duplicate processing, and an explicit reproduction of the out-of-order-request limitation above. Reproducing that limitation is not reported as resistance to it. These are software reference checks, not UAV hardware measurements.

**Table 3.** Transition-level protocol checks. The identifier of each row is the name of the test function in `test_protocol.c` of the supplementary materials.

| ID | Injected Event | Required Observation | Result |
|---|---|---|---|
| T1 | Deliver all five link transmissions. | UAV and backend finish with the same new state and session key. | Pass |
| T2 | Drop the wireless relay of $M_4$, then retry under a fresh challenge. | The old UAV state matches the backend previous slot and the pair resynchronizes. | Pass |
| T3 | Drop three consecutive $M_4$ relays. | The previous slot stays pinned to the UAV's unchanged credential; recovery still succeeds. | Pass |
| T4 | Deliver an old $M_4$ after timeout and after a new transaction has started. | The explicit $N_E$ comparison fails and no persistent state changes. | Pass |
| T5 | Replay an accepted $M_2$ verbatim. | The replay cache rejects the tuple. | Pass |
| T6 | Pair an old $M_2$ with a fresh $N_E$. | HMAC verification fails because $Auth_U$ covers $N_E$. | Pass |
| T7 | Flip one bit of $Auth_U$. | Constant-time verification fails and no state changes. | Pass |
| T8 | Flip one bit of $Auth_S$. | Constant-time verification fails and the UAV keeps its old credential. | Pass |
| T9 | Deliver one $M_3$ twice concurrently. | Exactly one request commits; the second observes the updated slot and fails. | Pass |
| T10 | Commit the later of two outstanding requests before the earlier one. | The earlier request still matches the previous slot and overwrites the current one; the UAV is left one state behind. Divergence reproduced as predicted in Section 5.4. | As predicted |

## 6. Security Analysis

This section combines the following authentication arguments with a Real-or-Random session-key analysis under the assumptions in Section 3. HMAC is treated as a secure pseudorandom function and unforgeable message-authentication code; nonces and keys have the lengths stated in Section 4.

### *6.1. Mutual Entity Authentication and Session-Key Agreement*

To make *MCC* accept, an adversary must produce $Auth_U$ for a fresh transcript under a key in the current or previous slot. Under HMAC unforgeability, this is infeasible without the corresponding $K_i$. To make $U_i$ accept, an adversary must produce the domain-separated $Auth_S$ under the pending $K_i$ for the same transcript. Backend acceptance establishes the

origin of a matching UAV request, while UAV acceptance establishes a matching backend processing event; neither implies final-message receipt at the other endpoint. The distinct tags prevent a valid output in one role from being reused as another role's authenticator or update value. When $M_4$ is delivered and verified, both endpoints derive the same $SK_i$ from the same canonical transcript.

### *6.2. Real-or-Random Session-Key Analysis*

We use a RoR-style experiment [46] for high-entropy shared keys. UAV and control-center instances are partnered by the same registered UAV record and canonical transcript $T = (PID_i, N_E, N_U)$. The control center accepts at atomic commit; the UAV accepts after server-token verification and local update. Execute exposes honest transcripts, Send permits active wireless delivery and modification, and Reveal exposes accepted session keys. The backhaul and gateways remain trusted as specified in Section 3. Corruption of a UAV excludes all its sessions, including earlier ones, from freshness. Thus the experiment does not claim forward secrecy.

Test is permitted for an accepted session whose key and partner key have not been revealed. All Test queries use one hidden bit $b$: for $b = 1$ the oracle returns the real 256-bit key, and for $b = 0$ it returns an independent uniform key for each distinct session identifier. Repeated and partner tests return consistent values. Reveal of either tested partner and subsequent corruption of a tested UAV are prohibited; Test on a previously corrupted UAV is rejected. Other session-key reveals remain allowed. Application traffic and side channels are outside this authentication-only game. Define $\mathrm{Adv}_{\mathcal{A}}^{RoR} = |2\Pr[b' = b] - 1|$.

Let $m$ bound the unexposed credential keys used, and $\epsilon_F$ bound each single-key HMAC PRF hybrid at the experiment's time and query budget [47]. Let $n$ bound the sampled or derived 128-bit keys, pseudonyms, and nonces, and $v$ bound adversarial token-verification attempts. A conservative bound, capped at one, is

$$\mathrm{Adv}_{\mathcal{A}}^{RoR} \le 2m\epsilon_F + 2\delta, \qquad \delta = \frac{n(n-1)}{2^{129}} + \frac{v}{2^{256}}. \tag{15}$$

To obtain it, start with the real game and replace HMAC under each unexposed credential key, in credential-generation order, by a consistent random function; the success-probability difference is at most $m\epsilon_F$. The distinct `KEY` domain makes evolved keys uniform in this hybrid. Reuse of a retained previous key uses its existing function table. Abort on a 128-bit collision or an origin-less successful token guess, with probability at most $\delta$. Collisions include hidden key collisions that could couple function tables. Outside these events, the output at $(\mathtt{SK}, T)$ is independent of the public authenticators and update outputs, which have distinct tags. Different fresh transcripts give distinct inputs; freshness excludes revelation through either partner. Replacing all tested keys by consistent random strings therefore leaves the view unchanged. The final guessing probability is $1/2$, and the triangle inequality yields (15). The same hybrids bound origin-less authentication acceptance by $m\epsilon_F + \delta$.

This argument relies on the revised $Auth_S = F_{K_i}(\mathtt{S}, T)$. A public token $F_{SK_i}(\mathtt{S}, T)$ would instead let an adversary test a candidate key by recomputing that token, defeating the stated RoR definition. The correction changes neither the number of HMAC evaluations nor message lengths. The result is conditional on PRF security and does not imply 256-bit overall security from a 128-bit credential. It establishes key secrecy and authentication correspondence, not delivery or general state-order safety; the symbolic analysis of Section 6.7 covers the flow properties that this argument leaves open.

### *6.3. Replay, Delay, and Man-in-the-Middle Resistance*

An old $M_2$ paired with a fresh gateway challenge fails because $Auth_U$ covers $N_E$. Reuse of an already accepted full tuple is rejected by the backend replay cache. An old $M_4$ cannot

update the UAV after timeout or during a later session because $N_E$ must equal the single active transaction. Modification of $PID_i$, $N_U$, $N_E$, or either authenticator changes an HMAC input or output and causes rejection. The protected backhaul prevents an attacker from replacing the gateway's pending $N_E$ in $M_3$.

### 6.4. Identity Privacy and Its Boundary

The permanent identity $ID_i$ never appears on the online channel. After a completed session, the next pseudonym is the truncated output of a keyed, domain-separated HMAC over fresh nonces; a passive observer without $K_i$ cannot compute the update relation. The resulting guarantee is conditional unlinkability across successfully completed updates. If an active adversary repeatedly blocks $M_4$, the UAV must reuse its old pseudonym to recover, so those failed attempts remain linkable. The protocol therefore does not claim unlinkability against an adversary that actively induces loss.

### 6.5. Desynchronization Resistance

Invariant I4 prevents the UAV from changing state before server authentication. If *MCC* advances and $M_4$ is lost, the accepted old state remains in $S_i^{pre}$. A fresh retry can therefore authenticate and derive a replacement current state. Explicit challenge binding prevents a delayed $M_4$ from updating a different active transcript, while invariant I3 serializes backend writes. Recovery is conditional on the request-ordering assumption and retention of the needed previous credential. The counterexample in Section 5.4 limits the claim for out-of-order requests. Backend state corruption or administrative rollback is outside the protocol model.

### 6.6. Key-Compromise and Availability Limits

This symmetric protocol does not claim standard perfect forward secrecy or post-compromise security. Extraction of the UAV's current secret enables future impersonation until re-enrollment or rekeying and may expose sessions associated with state still retained at either endpoint. One-way key updates and secure deletion can limit exposure of older discarded states, but that property is not presented as a formal forward-secrecy proof. Similarly, compromised gateways can deny service or expose association metadata, and wireless jamming remains an availability attack [48]. PUF protection, tamper-resistant storage, periodic rekeying, and multi-gateway monitoring are possible extensions rather than guarantees of the present protocol.

### 6.7. Automated Verification

The RoR argument above establishes key indistinguishability, but it does not by itself exclude a flow-level error such as accepting a response under the wrong transcript or committing two states for one request. We therefore model the protocol in Tamarin [49] under the Dolev–Yao adversary, which controls the wireless link and can eavesdrop, inject, modify, delay, and replay, while treating HMAC-SHA-256 as a perfect pseudorandom function. This is the symbolic counterpart of the computational assumption in Section 6.2: one analysis governs key strength, the other governs state and message flow.

Model.

The model defines eight rule roles: registration, gateway challenge, gateway forwarding, UAV request, one bounded recovery retry, UAV acceptance, backend verification against the current slot, and backend verification against the previous slot. Because the UAV-side and current-slot roles are instantiated once per credential epoch, the file contains twelve rules in total. Persistent facts carry the UAV credential, the pending transaction, and the

two backend slots. The replay cache is encoded as a single restriction stating that a given $(PID_i, N_E, N_U)$ tuple is accepted at most once, which is the property the cache must guarantee and is stronger than any particular eviction policy.

Bound.

A credential that evolves on every accepted request makes the term depth unbounded, and the proof search then diverges; bounding the number of epochs by a restriction does not help, because the state graph remains cyclic. The model therefore advances the credential through explicitly named facts for two epochs, which is exactly the span the recovery argument needs: one commit whose response is lost, followed by one successful recovery. Claims about an unbounded number of consecutive losses rest on the inductive argument of Section 5.4, not on the model.

Results.

Table 4 lists the five lemmas. Four are universally quantified; `recovery_exists` is an existential query that rules out the vacuous case in which the recovery path is unreachable and the universal claims therefore hold trivially. The full model and its verification log are provided as Supplementary File S2.

**Table 4.** Tamarin verification results. Step counts are those reported by the tool and are given for reproducibility.

| Lemma | Property Established | Steps |
|---|---|---|
| `sk_secrecy` | The adversary never derives $SK_i$. | 123 |
| `uav_agreement` | Every UAV acceptance has a prior commit on the same transcript and key. | 56 |
| `uav_agreement_injective` | One transcript is accepted at most once. | 56 |
| `mcc_agreement` | Every commit has a prior UAV request as predecessor. | 97 |
| `recovery_exists` | The loss-recovery path is reachable. | 15 |

Tool choice.

ProVerif would be the lighter option for a stateless protocol, but its table construct cannot delete rows in the version we used, so a credential that is replaced on every commit cannot be represented; the backend state would have to be over-approximated, which is precisely the behavior under test. Tamarin's multiset-rewrite rules express slot replacement directly, at the cost of the epoch bound noted above.

Scope.

The model covers message flow and state transitions. It does not cover implementation defects, timing side channels, random-number failures, or the computational strength of HMAC, all of which lie outside a symbolic analysis. Section 7.4 and Table 3 address the implementation layer separately.

## 7. Performance Evaluation

### *7.1. Evaluation Method*

The analysis counts operations and messages in the online mutual-authentication phase, as shown in Table 5. Let $T_h$ denote one SHA-256 evaluation over the 55-octet encoding of Section 4, $T_F$ one HMAC-SHA-256 evaluation over the same input, and $T_R$ one secure 128-bit nonce generation. Encoding, fixed-length concatenation, comparison, database lookup, and 128-bit truncation are not assigned artificial times. Communication uses 128-bit pseudonyms and nonces and 256-bit authenticators.

**Table 5.** Measured primitive costs on the Raspberry Pi Zero 2 W. Means of 1000 runs, net of the 1.33 µs no-op harness cost. The AEGIS and fuzzy-extractor rows timed a matched pair and are halved here.

| Operation | Symbol | Time (µs) | Backing |
|---|---|---|---|
| SHA-256, 32-octet input | $T_h$ | 8.8 | C library |
| HMAC-SHA-256, 32-octet input | $T_F$ | 36.7 | C library |
| 128-bit nonce | $T_R$ | 3.4 | CSPRNG |
| AEGIS-128L, 32-octet input | $T_{AG}$ | 57.0 | C library |
| Fuzzy extractor, Gen or Rep | $T_{FE}$ | 35.0 | hash commitment |
| Noisy XOR-PUF | $T_{PUF}$ | 2104.1 | pure Python |
| ECC scalar multiplication | $T_{G/R}$ | 5416.2 | pure Python |
| XOR, concatenation | $T_x$, $T_{cat}$ | 0 | — |

Measurement platform.

A count is exact; a time is not, because it depends on the platform and on the primitive library. Rather than combine figures from five testbeds, we measured every primitive the comparison needs on one board and priced all six protocols with that single set.

The board is a Raspberry Pi Zero 2 W Rev 1.0: a quad-core Cortex-A53 at 1.0 GHz (observed at 0.7 GHz, no throttling), 415 MiB of RAM, Debian 13 for AArch64, kernel 6.18, CPython 3.13.5, die temperature 46.2 °C, and a measured board power of 2.93 W. Each primitive was invoked 1000 times after a discarded warm-up, and the run also timed a no-op call to quantify the harness itself at 1.33 µs; the values below are means net of that overhead. Two rows time a pair of operations rather than one—AEGIS-128L encrypt followed by decrypt, and fuzzy-extractor Gen followed by Rep—and are halved, as indicated in the caption.

Two properties of the measurement matter for what follows. First, the protocol hashes 55 octets where the benchmark hashed 32, but both fit in a single 64-octet compression block once padding and the length field are added, so the measured $T_h$ and $T_F$ apply unchanged to the protocol transcript. Second, the measured ratio $T_F/T_h = 36.7/8.8 \approx 4.2$ is what the block structure predicts, since HMAC hashes two inner and two outer blocks; an implementation that disagreed with this ratio would be signalling a defect in one of the two rows.

The four primitives our protocol uses are all C-library calls, so its measured cost carries no interpreter penalty beyond the call boundary. The ECC and PUF rows are pure-Python research implementations and are correspondingly slower than a native library would be; they are used as measured, because substituting published figures from other boards would reintroduce exactly the cross-platform mixing this section is meant to remove.

### *7.2. Protocol-Design Characteristics*

Table 6 summarizes the architectural and primitive differences needed to interpret the later cost comparison. A broad yes/no security ranking is avoided because the protocols adopt different threat models and proof methods.

**Table 6.** Design characteristics of the compared protocols.

| Protocol | Authentication Path | Drone/UAV-Side Primitives | Link Transmissions | Freshness and State Handling |
|---|---|---|---|---|
| TCALAS [14] | User–GSS–drone | Hash operations | 3 | Temporal credentials, nonces, and timestamps |
| LAKE-IoD [15] | Mobile user–MS–drone | SHA-256 and AEGIS generation | 3 | Random nonces and timestamps |

**Table 6.** *Cont.*

| Protocol | Authentication Path | Drone/UAV-Side Primitives | Link Transmissions | Freshness and State Handling |
|---|---|---|---|---|
| SecAuthUAV [16] | UAV–GS; extended UAV–UAV | PUF, hash, XOR, nonce generation | 3 | Nonce freshness; GS retains new and previous PUF CRPs |
| SLAP-IoD [39] | Mobile user–control server–drone | Fuzzy extractor/PUF and hash | 4 | Random nonces, timestamps, and a PUF CRP |
| PSLAPS-IoD [17] | Drone–GSS–mobile device | Fuzzy extractor, hash, XOR, random generation | 4 | Timestamps; no synchronized credential replacement in the online phase |
| Proposed | UAV–edge gateway–MCC | HMAC-SHA-256 and nonce generation | 5 | Nonce-bound active transaction and atomic current/previous slots |

### *7.3. Computation Cost*

The UAV evaluates $F$ for $Auth_U$, $SK_i$, the expected $Auth_S$, $PID_i'$, and $K_i'$ and generates $N_U$. The backend performs the corresponding five HMAC evaluations, while the gateway generates $N_E$. Hence,

$$C_{UAV} = T_R + 5T_F, \tag{16}$$

$$C_{MCC} = 5T_F, \tag{17}$$

$$C_{EG} = T_R, \tag{18}$$

$$C_{total} = 2T_R + 10T_F. \tag{19}$$

These expressions count only the online authentication operations specified here. In particular, $C_{EG} = T_R$ and $C_{total} = 2T_R + 10T_F$ exclude the cryptographic work required to establish and maintain the protected *EG–MCC* channel, including its peer authentication, encryption/decryption, integrity protection, and replay checks. The channel remains a required deployment assumption rather than a zero-cost mechanism; its implementation-dependent setup and per-record costs must be added when estimating deployment cost.

Table 7 reports three things per protocol: the operation count transcribed from its source, the time the authors of that source measured on their own platform, and the time the same count costs when priced with the primitives of Table 5. The third column is the controlled comparison; the second is what each author actually observed, and the two are kept side by side because a single number would hide either the platform difference or the design difference.

The one assumption that moves the ranking.

The two measured columns differ in a single input: the cost of a fuzzy extractor. The extractor on our board is a hash-based fuzzy commitment—a salt, one masked codeword, and a Hamming check—which costs 35.0 µs per operation. The extractors in TCALAS, LAKE-IoD and SLAP-IoD are BCH constructions whose decoding dominates their published totals: TCALAS reports $30T_h + T_{FE} = 26.7$ ms with $T_h = 0.32$ ms, which leaves $T_{FE} \approx 17.1$ ms, roughly five hundred times our value. Substituting our commitment into their rows would therefore understate them by two orders of magnitude, and we report both bases rather than choose one.

Where the proposed protocol wins and where it does not.

On the UAV side the proposed protocol costs 186.9 µs, which is fourth of six under the hash-extractor basis and third under the BCH basis. It is never the cheapest on this

side, and the reason is visible in the counts: TCALAS evaluates seven hashes (62.0 µs), SLAP-IoD one extractor plus eight hashes (105.8 µs), and LAKE-IoD three hashes and two AEGIS calls (140.6 µs), whereas the proposed protocol evaluates five HMACs, each costing about four hashes. In total the picture depends on the extractor: at the measured cost of our own commitment the proposed protocol is third (373.9 µs, behind TCALAS at 300.5 µs and SLAP-IoD at 326.7 µs and ahead of LAKE-IoD at 474.6 µs), while at a BCH-class extractor cost it is the cheapest in the set, 15 times below PSLAPS-IoD.

**Table 7.** Computation cost. The count column is transcribed from each source; the two measured columns evaluate those counts with the primitives of Table 5, once with the hash-based fuzzy commitment measured here and once with a BCH-class extractor priced at 17.1 ms (see the text). Values are µs, UAV side / total.

| Protocol | Operation Count | Measured, Hash FE | Measured, BCH FE | Reported by Source |
|---|---|---|---|---|
| TCALAS [14] | $7T_h$ /$30T_h + T_{FE}$ | 62.0/300.5 | 62.0/17,365.5 | 2.24/26.7 ms |
| LAKE-IoD [15] | $3T_h + 2T_{AG}$/$11T_h + 6T_{AG} + T_{FE}$ | 140.6/474.6 | 140.6/17,539.6 | 0.205/0.894 ms [a] |
| SecAuthUAV [16] | $8T_x + 2T_R + 3T_h + 2T_{PUF} + 11T_{cat}$ | 4241.4/4241.4 | 4241.4/4241.4 | 31.7 µs/— |
| SLAP-IoD [39] | $T_{FE} + 8T_h$/$2T_{FE} + 29T_h$ | 105.8/326.7 | 17,170.8/34,456.7 | 5.320/12.371 ms |
| PSLAPS-IoD [17] | $9T_h + T_{G/R} + T_x + T_R$ / $23T_h + T_{G/R} + 3T_x + T_R$ | 5499.2/5623.1 | 5499.2/5623.1 | 4.431/4.504 ms [a] |
| Proposed | $T_R + 5T_F$/$2T_R + 10T_F$ | 186.9/373.9 | 186.9/373.9 | — |

$T_h$ is SHA-256. Symbols are normalized: LAKE-IoD prints $T_{SH}$ for SHA-256 and $T_{BU}$ for its fuzzy extractor, and PSLAPS-IoD prints $T_N$ for the nonce; they appear here as $T_h$, $T_{FE}$ and $T_R$. [a] Each source prints a total that its own constants do not reproduce—LAKE-IoD prints 0.8943 ms where its constants give 0.9254 ms, PSLAPS-IoD prints 4.504 ms where its constants give 4.4948 ms—and the printed values are shown. SecAuthUAV instantiates SHA-1 rather than SHA-256, so its measured column uses a hash function it does not deploy and is indicative only. PSLAPS-IoD defines $T_{G/R}$ only as "generation and replication functions"; it is priced as one ECC scalar multiplication, the most favorable reading for that protocol.

### *7.4. Reference Implementation*

The four phases are implemented as dependency-free C99 over explicit state: a credential slot at the UAV, a pending-transaction record, a two-slot backend record, and a replay cache. SHA-256 and HMAC-SHA-256 live in the same translation unit and reproduce the FIPS 180-4 and RFC 4231 test vectors, so the footprint below is what the protocol actually requires rather than a linked but unused library. The ten checks of Section 5.5 run from the same build, and the encoding used for every HMAC input is the 55-octet encoding fixed in Section 4.

Memory.

The protocol and cryptographic translation units occupy 3.5 KB of text. The UAV-resident state is 88 bytes: a pseudonym, an evolving key, a pending-transaction record, and the flags that guard it. The backend keeps 3156 bytes per registered UAV, of which 3072 bytes are the 64-entry replay cache; the remainder is the current and previous credential slots. All three figures are compile-time constants of the released code and carry over unchanged to a target of the same word size.

What the measured cost includes.

The measured total of 373.9 µs is almost entirely the ten HMAC evaluations and two nonce generations: $10T_F + 2T_R$ already accounts for it, and the state-machine logic around those calls adds roughly two per cent. That is consistent with a thin implementation, and it also means the figure can be reproduced from Table 5 without running the code.

Two limits apply. The figures are CPython-level, so a C implementation of the same primitives would be faster in absolute terms; because the per-call penalty is fixed, a port to C would help designs with many evaluations more than it would help this one, which has ten. And the C reference implementation of Section 5.5 runs the same ten HMAC evaluations in 6.34 µs on a 3.2 GHz desktop core, so the gap between that figure and 373.9 µs is the combined effect of clock rate, microarchitecture and interpreter, not of the protocol.

Energy.

The board draws 2.93 W, so an authentication that occupies 373.9 µs consumes 1095 µJ at the board level, of which the UAV side accounts for 548 µJ. This is the energy the board uses during the interval, not the marginal energy attributable to the computation: a board of this class draws most of its power whether or not it is hashing, so the figure is an upper bound on what the protocol costs in energy and should be read together with the idle draw. Radio energy for the 1024-bit UAV-facing payload is additional and is not included.

### *7.5. Communication Cost*

For the proposed messages,

$$|M_1| = |N_E| = 128, \tag{20}$$

$$|M_2| = |PID_i| + |N_U| + |Auth_U| = 512, \tag{21}$$

$$|M_3| = |N_E| + |M_2| = 640, \tag{22}$$

$$|M_4| = |N_E| + |Auth_S| = 384 \quad \text{bits.} \tag{23}$$

The UAV-facing radio traffic is

$$C_{radio} = |M_1| + |M_2| + |M_4| = 1024 \text{ bits.} \tag{24}$$

The protected backhaul carries $M_3$ and one copy of $M_4$, also 1024 bits of authentication protocol payload. The aggregate authentication protocol payload is therefore

$$C_{all\ links} = 1024 + 1024 = 2048 \text{ bits} \tag{25}$$

over five link transmissions. These counts include only the authentication message fields. They exclude link/transport headers, protected-backhaul cryptographic overhead (such as record tags, nonces, and channel-establishment traffic), and retransmissions; they are not complete on-wire traffic measurements. The HMAC input encoding defined in Section 4 is local computation and adds no transmitted fields.

Table 8 separates two views. Panel A compares the proposed all-link authentication payload with the published protocol totals, accounting for architectural differences. Panel B presents the link-wise decomposition of the proposed protocol. The 1024-bit radio payload is a component of the 2048-bit total and is not directly compared with the baseline totals.

**Table 8.** Authentication payload comparison and separate proposed-link breakdown.

| **Panel A. Published totals and proposed all-link total** | | |
|---|---|---|
| **Protocol** | **Link Transmissions** | **Payload** |
| TCALAS [14] | 3 | 1536 bits |
| LAKE-IoD [15] | 3 | 1376 bits |
| SecAuthUAV [16] | 3 | 1600 bits |
| SLAP-IoD [39] | 4 | 2048 bits |
| PSLAPS-IoD [17] | 4 | 1728 bits |
| Proposed, all links | 5 | 2048 bits |

**Table 8.** *Cont.*

| **Panel B. Proposed protocol only, components of the total** | | |
|---|---|---|
| **Link Scope** | **Link Transmissions** | **Payload** |
| Proposed, UAV radio | 3 | 1024 bits |
| Proposed, protected backhaul | 2 | 1024 bits |

Panel B decomposes the proposed total in Panel A; its radio-only value is not a baseline comparison. Proposed values exclude link/transport headers, protected-backhaul cryptographic overhead, and retransmissions. Baseline values retain their published accounting.

### *7.6. Storage Cost*

Table 9 reports persistent credential storage: 384 bits at the UAV and 640 bits per UAV at *MCC*. Replay-cache entries, indexes, pending transcripts, and session keys are volatile or deployment-dependent and are excluded.

**Table 9.** Persistent storage of the proposed protocol.

| **Entity** | **Persistent Values** | **Cost** |
|---|---|---|
| UAV | $ID_i, PID_i, K_i$ | 384 bits |
| Edge gateway | Site credential only; no UAV secret | Variable |
| Control center | $ID_i, S_i^{cur}, S_i^{pre}$ | 640 bits/UAV |

### *7.7. Discussion*

The measured costs are platform-specific in a way that the counts are not: they follow from Table 5, and a different library would move every row. Two conclusions follow from the table and they point in opposite directions. The UAV-side figure, 186.9 µs, is not competitive with the hash-only designs, and the revision states that rather than avoids it. The total figure depends on how the fuzzy extractor is priced: with the hash commitment measured here the protocol is third at 373.9 µs, and with a BCH construction at the cost TCALAS's own numbers imply it is the cheapest in the set. The claim to lightness therefore rests on avoiding two primitive classes, on a constant ten evaluations of a single symmetric primitive, and on 88 bytes of UAV state, not on being the fastest implementation in either column.

The communication picture is similarly mixed. The all-link authentication payload is 2048 bits, equal to the published SLAP-IoD total and larger than the other four, because the proposed architecture forwards both request and response across a separate backhaul; the 1024-bit radio component describes only the UAV-facing link and does not establish an advantage over baseline totals. The trade-off is also security-specific: the protocol does not provide hardware unclonability, standard perfect forward secrecy, post-compromise recovery, or radio availability. Its contribution is a small HMAC-based state machine with explicit delayed-message rejection and one-step loss recovery for a trusted mine backend, under the stated request-ordering condition, verified by ten transition checks and a symbolic model rather than asserted.

*Extreme threat boundaries.* If the *EG* is fully compromised, the adversary can replace its generated $N_E$, drop or delay $M_4$, and observe all wireless metadata; however, because the *EG* stores no $K_i$ and cannot forge $Auth_U$ or $Auth_S$, it cannot impersonate an arbitrary UAV to the *MCC* or forge server responses. Its compromise therefore degrades availability and metadata privacy, not the unforgeability of the protocol. Large-scale DDoS against the *MCC* forces concurrent per-UAV atomic transactions and can induce row-lock contention on the pseudonym index; mitigations include rate-limiting at the *EG*, stateless challenge pre-validation, an in-memory replay-cache layer, and sharding the credential table by pseudonym prefix.

## 8. Conclusions

This paper presented a nonce-bound lightweight authentication protocol for a UAV, an edge gateway, and a mine control center. HMAC domain separation, explicit challenge binding, one-active-transaction semantics, atomic backend updates, and current/previous credential slots address replay, delayed response, message modification, and final-response loss under the stated trust model and backend request-ordering condition. Transition tests exercised the principal state paths, and the performance evaluation compared computation and communication costs with five published protocols. The proposed UAV cost is $T_R + 5T_F$; its authentication protocol payload is 1024 bits on the radio link and 2048 bits across all links, excluding link/transport headers and protected-backhaul cryptographic overhead. Future work should implement the protocol on representative UAV hardware, measure energy and latency with a common cryptographic library, and investigate capture-resistant key storage and recovery after endpoint compromise.

**Supplementary Materials:** The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/electronics15225370/s1, File S1, the C99 reference implementation (`crypto.c`, `protocol.c`, `proto.h`), the ten checks (`test_protocol.c`), the benchmark harness (`bench.c`) and a Makefile; File S2, the Tamarin model (`uav_mine_r3.spthy`) and its verification log.

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

**Funding:** This research was funded by the National Science Foundation of China (Grant No. 42230816) and the National Science and Technology Major Project on New Oil and Gas Exploration and Development (Grant No. 2025ZD1400505), and Research Project of Tarim Oilfield (Grant No. 041024110221). The authors extend their appreciation to the Deanship of Scientific Research and Graduate Studies at King Khalid University, KSA, for funding this work through Small Research Group under grant number RGP.1/39/46.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The C99 reference implementation, the ten transition-level checks of Table 3, the benchmark harness, and the Tamarin model together with its verification log are provided in Supplementary Materials (Files S1 and S2). All other data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Park, S.; Choi, Y. Applications of Unmanned Aerial Vehicles in Mining from Exploration to Reclamation: A Review. *Minerals* **2020**, *10*, 663. [CrossRef]
2. Dang, T.M.; Nguyen, B.D. Applications of UAVs in Mine Industry: A Scoping Review. *J. Sustain. Min.* **2023**, *22*, 128–146. [CrossRef]
3. Zhang, Z.; Zhu, L. A Review on Unmanned Aerial Vehicle Remote Sensing: Platforms, Sensors, Data Processing Methods, and Applications. *Drones* **2023**, *7*, 398. [CrossRef]
4. Tong, X.; Liu, X.; Chen, P.; Liu, S.; Luan, K.; Li, L.; Liu, S.; Liu, X.; Xie, H.; Jin, Y.; et al. Integration of UAV-based photogrammetry and terrestrial laser scanning for the three-dimensional mapping and monitoring of open-pit mine areas. *Remote Sens.* **2015**, *7*, 6635–6662. [CrossRef]
5. Stothard, P.; Shirani Faradonbeh, R. Application of UAVs in the Mining Industry and Towards an Integrated UAV-AI-MR Technology for Mine Rehabilitation Surveillance. *Min. Technol. Trans. Inst. Min. Metall.* **2023**, *132*, 65–88. [CrossRef]
6. Yin, Z.; Bi, Z.; Zhang, F. Multi-Objective Edge Node Deployment Method Based on Improved Heuristic Algorithms for Smart Mines. *Appl. Sci.* **2024**, *14*, 10903. [CrossRef]

7. Sun, G.; Wang, Y.; Yu, H.; Guizani, M. Proportional Fairness-Aware Task Scheduling in Space-Air-Ground Integrated Networks. *IEEE Trans. Serv. Comput.* **2024**, *17*, 4125–4137. [CrossRef]
8. Lei, T.; Mao, K.; Zhu, Q.; Zhang, R.; Ma, Z.; Hua, B.; Niyato, D.; Wu, Q. Ultra-wideband channel modeling and characterization under low altitude scenarios. *Chin. J. Aeronaut.* **2026**, *39*, 104077. [CrossRef]
9. Wang, T.; Zhu, Z.; Zhou, X.; Jing, T.; Chen, W. A Function-Based Behavioral Modeling Method for Air Combat Simulation. *J. Syst. Eng. Electron.* **2024**, *35*, 945–954. [CrossRef]
10. Tang, H.; Bai, C.; Guo, J. HAPPD: Cooperative Pursuit Strategy for Fixed-Wing UAVs in Complex Terrain. *IEEE Trans. Autom. Sci. Eng.* **2026**, *23*, 9076–9091. [CrossRef]
11. Su, C.; Yuan, J.; Zheng, E.; Xu, W.; Liu, Z.; Hu, J. CPFL: Resilient Continuous UAV Localization via Cross-View Perception and Particle Filtering. *Drones* **2026**, *10*, 437. [CrossRef]
12. Yahuza, M.; Idris, M.Y.I.; Ahmedy, I.B.; Wahab, A.W.A.; Nandy, T.; Noor, N.M.; Bala, A. Internet of Drones Security and Privacy Issues: Taxonomy and Open Challenges. *IEEE Access* **2021**, *9*, 57243–57270. [CrossRef]
13. Yang, W.; Wang, S.; Yin, X.; Wang, X.; Hu, J. A Review on Security Issues and Solutions of the Internet of Drones. *IEEE Open J. Comput. Soc.* **2022**, *3*, 96–110. [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. Tanveer, M.; Zahid, A.H.; Ahmad, M.; Baz, A.; Alhakami, H. LAKE-IoD: Lightweight Authenticated Key Exchange Protocol for the Internet of Drone Environment. *IEEE Access* **2020**, *8*, 155645–155659. [CrossRef]
16. Alladi, T.; Naren; Bansal, G.; Chamola, V.; Guizani, M. SecAuthUAV: A Novel Authentication Scheme for UAV-Ground Station and UAV-UAV Communication. *IEEE Trans. Veh. Technol.* **2020**, *69*, 15068–15077. [CrossRef]
17. Algarni, F.; Jan, S.U. PSLAPS-IoD: A Provable Secure and Lightweight Authentication Protocol for Securing Internet-of-Drones (IoD) Environment. *IEEE Access* **2024**, *12*, 45948–45960. [CrossRef]
18. Qiu, H.; Hai, X.; Zhang, X.; Wang, Y.; Li, K.; Wen, C. Bio-Inspired Learning for Multi-UAV Cooperative Maneuvering Decisions Under Communication Denial. *IEEE Trans. Aerosp. Electron. Syst.* **2026**, *62*, 20243–20256. [CrossRef]
19. Yin, Y.; Yin, Y.; Wei, Y.; Su, Q.; Li, J. Dynamic Trend Perception for Fine-Grained Unmanned Aerial Vehicle Anomaly Detection in Flight Data. *Eng. Inf. Technol. Electron. Eng.* **2026**, *27*, 1–16. [CrossRef]
20. Liu, B.; Guo, J.; Wang, Y.; Yang, C.; Nan, F.; Yang, Y. Intelligent Inspection of Electronic Devices in Specific Environments via a Novel Cascade Network of Combining Mixed Sampling and Nonstrided Convolution. *IEEE Trans. Syst. Man Cybern. Syst.* **2025**, *55*, 3287–3299. [CrossRef]
21. Wang, Q.; Zhang, Y.; Fan, S.; Zhu, H.; Wei, J.; Gao, W. A Novel Cooperative Positioning Algorithm Based on High-Precision IMU Preintegration and Adaptive Sliding Observation Optimization. *IEEE Trans. Instrum. Meas.* **2026**, *75*, 9525312 [CrossRef]
22. Huang, X.; Yuan, J.; Tian, C.; Chen, N.; Qi, H.; Tan, A.; Zheng, E. HTGL: UAV Visual Geo-Localization Network with Transformer and Hypergraph Feature Aggregation Enhancement. *Drones* **2026**, *10*, 488. [CrossRef]
23. Huang, F.; Yu, Y.; Feng, T. Automatic building change image quality assessment in high resolution remote sensing based on deep learning. *J. Vis. Commun. Image Represent.* **2019**, *63*, 102585. [CrossRef]
24. Han, F.; Wang, Z.; Feng, Y.; Huang, J.; Li, M.; Yang, P.; Fu, Z. Use Your PIN Securely: Acoustic-Based Second-Factor User Authentication with Enhanced Tap Biometrics. *IEEE Trans. Dependable Secur. Comput.* **2026**, *23*, 11116–11133. [CrossRef]
25. Li, Y.; Lu, Z.; Deng, J.; Yao, Y.; Yin, Y.; Li, G.; Gu, S.; Peng, Z.; Xiao, B. Multi-Agent Reinforcement Learning Assisted Blockchain System for Dynamic UAV Task Allocation. *IEEE Internet Things J.* **2026**, *13*, 51858–51874 [CrossRef]
26. Tan, H.; Gu, S.; Zhu, H.; Shen, J.; Bhuiyan, Z.A.; Wu, Q.M.J. An Efficient ASCON-Based Group Authentication and Key Agreement Scheme with Non-Linkability and Integrity Assurance for IIoTs. *IEEE Trans. Dependable Secur. Comput.* **2026**, *23*, 8032–8047. [CrossRef]
27. Sourav; Ali, R.; Obaidat, M.S. Lattice-based Certificateless Signcryption Scheme using Blockchain for Smart Healthcare Systems. *IEEE Trans. Emerg. Top. Comput.* **2026**, 1–12. [CrossRef]
28. Zhan, X.; Zhang, X.; Wang, X.; Diao, X.; Qi, L. Comparative Analysis of Surface Deformation Monitoring in a Mining Area Based on UAV-LiDAR and UAV Photogrammetry. *Photogramm. Rec.* **2024**, *39*, 373–391. [CrossRef]
29. Yang, P.; Esmaeili, K.; Goodfellow, S.; Ordóñez Calderón, J.C. Mine Pit Wall Geological Mapping Using UAV-Based RGB Imaging and Unsupervised Learning. *Remote Sens.* **2023**, *15*, 1641. [CrossRef]
30. Hao, J.; Zhang, X.; Wang, C.; Wang, H.; Wang, H. Application of UAV digital photogrammetry in geological investigation and stability evaluation of high-steep mine rock slope. *Drones* **2023**, *7*, 198. [CrossRef]
31. Turner, D.; Lucieer, A.; De Jong, S. Time series analysis of landslide dynamics using an unmanned aerial vehicle (UAV). *Remote Sens.* **2015**, *7*, 1736–1757. [CrossRef]
32. Samaei, M.; Stothard, P.; Shirani Faradonbeh, R.; Topal, E.; Jang, H. Mine Closure Surveillance and Feasibility of UAV–AI–MR Technology: A Review Study. *Minerals* **2024**, *14*, 110. [CrossRef]

33. Meng, X.; Zhang, D.; Dong, S.; Yao, C. Open-Pit Granite Mining Area Extraction Using UAV Aerial Images and the Novel GIPNet. *Remote Sens.* **2024**, *16*, 789. [CrossRef]
34. He, Y.; Lai, Y.; Chen, B.; Chen, Y.; Xie, Z.; Yu, X.; Luo, M. An Open-Pit Mines Land Use Classification Method Based on Random Forest Using UAV: A Case Study of a Ceramic Clay Mine. *Minerals* **2024**, *14*, 1282. [CrossRef]
35. Don, M.G.; Wanasinghe, T.R.; Gosine, R.G.; Warrian, P.J. Digital Twins and Enabling Technology Applications in Mining: Research Trends, Opportunities, and Challenges. *IEEE Access* **2025**, *13*, 6945–6963. [CrossRef]
36. Jin, L.; Zhang, Y.; Li, Y.; Wang, S.; Yang, H.H.; Wu, J.; Zhang, M. MoE2: Optimizing Collaborative Inference for Edge Large Language Models. *IEEE Trans. Netw.* **2026**, *34*, 4637–4651. [CrossRef]
37. Zhang, H.; Zhang, D.; Wen, J.; Shen, S.; Fu, L.; Liu, W.; Li, E.; Song, A. A Novel FTP+d Control Approach With Event-Driven Communication for Time-Delayed Bilateral Teleoperation Systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2026**, *56*, 428–442. [CrossRef]
38. Yu, H.; Song, H.; Chi, G.; Xu, Z.; Gao, Y.; Liang, Q.; Hou, M.; Shi, Z.; Deng, J. Arsenic substitution and galvanic coupling in pyrite trigger uranium mineralization in sandstone. *Commun. Earth Environ.* **2026**, *7*, 624. [CrossRef]
39. Yu, S.; Das, A.K.; Park, Y.; Lorenz, P. SLAP-IoD: Secure and Lightweight Authentication Protocol Using Physical Unclonable Functions for Internet of Drones in Smart City Environments. *IEEE Trans. Veh. Technol.* **2022**, *71*, 10374–10388. [CrossRef]
40. Huang, F.; Qi, X.; Li, C.; Hu, W. Aerial image classification by learning quality-aware spatial pyramid model. *Future Gener. Comput. Syst.* **2020**, *111*, 271–277. [CrossRef]
41. Wang, B.; Zhao, H.; Hu, X.; Shen, Y.; Li, N. Adaptive sliding mode fault-tolerant control of an over-actuated hybrid VTOL fixed-wing UAV under transition flight. *ISA Trans.* **2026**, *169*, 263–273. [CrossRef]
42. Ma, X.; Kang, Y.; Fan, Y.; Dong, X.; Zhao, H. Intelligent identification and mechanism analysis of kaolinite-bearing rocks from coal gangue via dual-energy X-ray and gradient boosting. *Miner. Eng.* **2026**, *242*, 110219. [CrossRef]
43. Liu, P.; Wei, A.; Zhang, C.; Chen, Z.; Feng, C.; Lu, F.; Hu, W.; Lu, K. DRDFNet: A Degradation-Aware Restoration and Detail-Preserving Fusion Network for Infrared and Visible Image. *IEEE Trans. Image Process.* **2026**, *35*, 6248–6263. [CrossRef]
44. Cao, K.; Wang, T.; Diamantoulakis, P.D.; Li, X.; Yuen, C.; Karagiannidis, G.K. Reliable and Secure Wireless-Powered Communications via Hybrid Active-Passive Double-RIS. *IEEE J. Sel. Areas Commun.* **2026**, *44*, 3828–3844. [CrossRef]
45. Bellare, M.; Canetti, R.; Krawczyk, H. Keying Hash Functions for Message Authentication. In *Proceedings of the Advances in Cryptology—CRYPTO '96*; Springer: Berlin/Heidelberg, Bermany, 1996; pp. 1–15. [CrossRef]
46. Abdalla, M.; Fouque, P.A.; Pointcheval, D. Password-Based Authenticated Key Exchange in the Three-Party Setting. In *Proceedings of the Public Key Cryptography—PKC 2005*; Springer: Berlin/Heidelberg, Bermany, 2005; pp. 65–84. [CrossRef]
47. Bellare, M. New Proofs for NMAC and HMAC: Security without Collision-Resistance. In *Proceedings of the Advances in Cryptology—CRYPTO 2006*; Springer: Berlin/Heidelberg, Bermany, 2006; pp. 602–619. [CrossRef]
48. Cao, K.; Xu, S.; Diamantoulakis, P.D.; Yang, C.; Zheng, B.; Li, X.; Yuen, C. Self-Sustainable Active Metasurface (SAM): Reliable and Secure Communications. *IEEE Trans. Wirel. Commun.* **2026**, *25*, 12325–12340. [CrossRef]
49. Meier, S.; Schmidt, B.; Cremers, C.; Basin, D. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Proceedings of the Computer Aided Verification (CAV 2013)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Bermany, 2013; Volume 8044, pp. 696–701. [CrossRef]


**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.