Ατομική Διπλωματική Εργασία

SECURE DATA TRANSMISSION FOR UAV IN ADVERSARIAL ENVIRONMENTS

Γεώργιος Κουππάρης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2026

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Secure data transmission for UAV in adversarial environments

Γεώργιος Κουππάρης

Επιβλέπων Καθηγητής
Βασος Βασιλειου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2026

Acknowledgements

I would like to express my warm thanks to my supervising professor, Dr. Vasos Vasileiou, for his unwavering support, guidance, and trust throughout the preparation of this diploma thesis. His valuable advice, constant availability, and genuine interest contributed decisively both to the approach of the topic and to the successful completion of this work. His contribution was particularly important to me, and I sincerely thank him.

I would like to express my sincere thanks to Dr. Panagiotis Kolios for his valuable guidance, support, and encouragement throughout the course. His insightful comments, academic advice, and willingness to assist whenever needed contributed significantly to the development and successful completion of this work. His support was greatly appreciated, and I am thankful for his contribution.

Abstract

This thesis defines a mission-focused, secure multi-link communication protocol for each stage of UAV missions, first establishing trust between the system and mission stakeholders, followed by deployment, secure termination, and system sanitization. The proposed architecture incorporates trust creation, mutual verification, mission initialization, compartmentalized traffic key definition, protected multi-link data transportation, anomaly detection, and secure mission termination. In Phase A, trust is established using a series of pseudonymous identities, a mission-scoped trust database, and secure operational policies. In Phase B, an Authentication Key Exchange (AKE) is established between the Ground Control Station (GCS) and UAV to provide mutual authentication and generate a new Session Master Key, achieving Perfect Forward Secrecy. Phase C sets up secure mission initialization and serves as explicit key confirmation, and Phase D converts the Session Master Key into independent traffic keys to attain strong cryptographic separation between directions, channels, and links. Phase E provides real-time flight operational communication through several protection mechanisms, including Ascon-AEAD128, All-Or-Nothing Transform (AONT), Reed-Solomon coding, anti-replay schemes, key rotation, and multi-link scheduling. Meanwhile, telemetry is processed by an anomaly detection algorithm based on an LSTM autoencoder for dynamic risk evaluation and adaptive trust management. Finally, Phase F provides secure mission termination through zeroization of volatile secrets, cryptographic erasure of mission data, and retention only of the necessary long-term identities in a secure environment.

Table of Contents

Chapter 1

Introduction

1.1 History of UAVs

Unmanned aerial vehicles (UAVs), also known as drones, are a significant part of modern aeronautics. A UAV is an aircraft without a human operator, which can be remotely controlled, semi-autonomous, or autonomous. While UAVs are frequently linked to fresh technology advancements, their history stretches back over 100 years. Military requirements, radio controllers, miniaturized electronic technologies, satellite navigation, digital communications technologies, more recently AI and autonomous task planning have all influenced the evolution of UAVs [1], [2]. Military developments were closely linked to the earliest concepts of UAVs. Both Britain and the United States designed early pilotless aircraft during World War I. Britain conducted tests of the Aerial Target in March 1917, a small radio-controlled aircraft, while the United States was also testing the Kettering Bug, an experimental unmanned aerial torpedo, which was developed in October 1918 [2], [3]. Although the systems were never used operationally during the war, they were early efforts to displace the pilot from the aircraft and control aerial platforms through mechanical, electrical or radio means. The UAV development continued during the interwar years and into World War II. These platforms were frequently used for anti-aircraft gunner training and flight testing in the air-defence system. With the evolution of those technical underpinnings, UAVs eventually surpassed the target capability to the ability to engage in reconnaissance, observation and intelligence collection [1], [4]. The Cold War was a turning point in the development of UAVs in the military field. Military operations evolved significantly, and as a result, intelligence collection became increasingly important without exposing pilots to actual threat [5]. In the later 20th century, improvements in computing, sensors, communications and navigation made UAVs much more powerful. The integration of

satellite and Global Positioning System communication technology (GPS) had enabled unmanned aerial vehicles to carry out long-range reconnaissance in realistic, challenging environments where it was dangerous to operate manned aircraft, with a high level of surveillance in real-time. These technologies also enhanced video surveillance, operational decisions, responded more quickly to threats and completed missions at times when conventional missions had less power and potential or were more hazardous, according to a historical study by the United States Air Force. Beginning in the first few decades of the 21st, UAVs moved from military applications to civilian, commercial, and scientific uses. This development is significant. Therefore, the history of UAVs is not only a story of aviation progress but also a story of increasing dependence on secure communications.

1.2 Applications of UAVs

UAVs are deployed in numerous sectors today because they provide rapid aerial observation, operational flexibility, reduced human risk, and the ability to manoeuvre through challenging terrain. Their value comes from the combination of mobility, sensor payloads, autonomous functions, and real-time communication. This scope of applications illustrates that UAVs have evolved into multifunctional cyber-physical platforms.

1.2.1 Precision Agriculture

Precision agriculture is one of the most well-known and lucrative applications of UAV technology in the civilian sphere. With aerial images collected, these methods help farmers and agricultural organizations identify crop health, assist with yield estimation and make spraying targeted for different areas and conditions as well. UAVs can perform rapid data collection over key sites of interest, facilitating the measurement of crop variability and the optimization of agricultural inputs [9], [11]. Such UAVs usually have RGB cameras, multispectral sensors, thermal cameras, or spraying payloads. Vegetation-index analysis using multispectral imagery, along with thermal measurements can help indicate irrigation stress. UAV systems may enable variable-rate spraying of pesticides, fertilizers, or herbicides, to reduce excessive chemical intervention and enhance productivity [11], [12]. In larger fields where it is inconvenient to gather information from manual examination, UAVs are essential. With early detection of issues relating to crop weakness, areas that lack water, or places where pests are an issue, UAVs can lead in timely decision-making and minimizing economic losses.

1.2.2 Building and Infrastructure Inspection

UAVs are also common for building and infrastructure inspection. They can inspect bridges, towers, rooftops, pipelines, power lines, wind turbines, industrial facilities, and so forth without workers having to pick up work at hazardous sites. The inspection risk is minimized, operational cost is reduced and infrastructure condition can be monitored more frequently. UAVs are especially valuable for structural inspection when traditional methods would require costly or disruptive equipment, such as scaffolding, cranes, helicopters, or temporary service interruptions [13], [14]. For infrastructure checks, UAVs possess high-resolution cameras, thermal sensors, and LiDAR units among other payloads. These payloads enable inspectors to look for the presence of cracks and corrosion, heat leakages, structural deformations, damaged parts, and other signs of a broken structure. Recent review papers as well have identified UAVs as safer, more effective, and economical alternative for the inspection purposes [13]. UAVs are important in this field because

infrastructure inspection is directly connected to public safety. Therefore, the communication system must be safeguarded, since these missions may involve sensitive critical-infrastructure information.

1.2.3 Traffic Monitoring and Analysis

In addition, UAVs are increasingly used for traffic monitoring and transport monitoring. Contrary to fixed cameras, UAVs can be set up on the scene dynamically, on incident scenes, temporary roadworks, major events, and in areas without permanent monitoring infrastructure. Studies on UAV use in road safety have shown their effectiveness for traffic observation, accident detection, road-condition monitoring, and transport-infrastructure studies [15]. Their view provides front-line traffic officials with aerial visibility of vehicle traffic, load, queue, lane and accident areas. This can increase situational awareness and help to reduce decision-making times in emergencies and during rush hour. UAVs can also be employed for studying road safety data regarding driving behaviour, transport congestion, pedestrian flow, and dangerous intersections. The operational implications for the traffic surveillance of the UAVs become heavily highlighted in the integration of ITS systems. In the event of this happening, the UAV data can be transmitted to traffic control centers, emergency response forces and automated analytics systems.

1.2.4 Disaster and Emergency Response

UAVs can deploy rapidly after earthquakes, floods, wildfires, landslides, or industrial accidents to collect aerial images and map out the disaster area, aid search-and-rescue operations and provide first responders with situational awareness. In a fire, flood or natural disaster situation, ground access may be blocked, infrastructure may be damaged, and human responders are put in danger. UAVs can therefore fill that critical information gap.[10], [16]. In such missions, UAVs take real-time photos in dangerous locations. For search-and-rescue missions, thermal camera-equipped UAVs can be employed to detect individuals during low-light or low-visibility operations. During wildfire response, UAVs have the capability to monitor fire spread, spot hotspots in a wildfire and conduct safer strategy planning for firefighting teams [16], [17]. There are also high levels of communication that UAVs in disaster response need to achieve. In case of communication failure or compromise, emergency staff could lose situational awareness.

1.2.5 Delivery and Logistics

UAVs are seen as a solution for a variety of delivery and logistics services such as lightweight packages, medical supplies, emergency gear and delivery. Their biggest benefit is that they can avoid congestion, reach isolated areas, and shorten delivery times when ground transportation is delayed or unavailable. This application transforms the UAV from a sensing instrument to a transport tool. In a mission like this, the UAV must secure its path and reliably communicate with operators. The increasing number of UAVs in shared low-altitude airspace makes the interoperability and interaction between UAVs, operators, and airspace-management systems more essential. [18]

1.2.6 Military Surveillance and Reconnaissance

Surveillance and reconnaissance are still among the most important uses in military contexts. UAVs are utilized to actively observe adversarial terrain, locate targets, assist border defence, enhance battlefield awareness and provide extended observation capability without putting pilots at risk of hostile fire. Historically, military need has been one of the main forces pushing the evolution of the unmanned aerial vehicle industry, including the Predator, Reaper, and Global Hawk [6], [7], [8], and thus have shown the operational

importance of long-endurance unmanned surveillance platforms. When UAVs are in military contexts, they usually work under highly adversarial conditions where their communication links are intentionally targeted. Military UAVs are also permitted to perform beyond visual line of sight, over long distances or against locations where satellite and radio communication networks are contested. Therefore, secure and robust communication is not optional.

1.3 Types of UAVs

UAVs can be classified according to airframe configuration, range, endurance, altitude, payload capability, propulsion method, mission role, and communication architecture. These categories are relevant because the design and operational profile of a UAV directly affect its flight behaviour, payload capacity, communication range, energy consumption, and security requirements. Not only are these categories mechanical or operational labels, but they also represent distinct mission profiles and communication-security requirements [19], [20].

1.3.1 Fixed Wing UAVs

Fixed-wing UAVs generate lift via wings much like traditional aircraft. The aerodynamic design allows them to fly at the best performance rate, which translates to better endurance, cruising speed, and operational range. That makes them suited to missions that force the UAV into the air for long expanses or long distances. Fixed wing UAVs typically have a higher payload and flight time as the power consumption during gliding flight is reduced. Fixed wing UAVs have been employed to investigate enormous fields such as border surveillance, farm watching, environmental monitoring, maritime monitoring, long-range reconnaissance, [23] [24]. High performance and energy efficiency is mainly the advantage of fixed-wing UAVs. However, fixed wing UAVs are also subject to operational limitations. For most fixed wing UAVs, either a runway or a stand-alone take-off and landing system will be required. This constraint affects mission design and communication behaviour as well.

1.3.2 Rotary Wing UAVs

A rotary-wing UAV generates lift from one or more rotating propellers. This includes single-rotor helicopters, quadcopters, hexacopters, octocopters and other types of multirotor configurations. Rotary-wing aerial vehicles are one of the most common classes of UAVs both in civilian and business. Unlike fixed-wing UAVs, rotary-wing vehicles can be mounted in an almost static posture, allowing them not only to survey particular areas but also inspect them stably. It makes them able to be employed for tasks requiring continuous observation [23]. Because the motors are always generating lift, their endurance and range are less than that of fixed-wing UAVs. They are preferable for local or medium-duration missions. Furthermore, the use of rotary-wing UAVs can be applied in busy, urban, industrial and emergency environments, where the communication channel is hindered by interference, multipath propagation, obstruction and signal degradation. As these UAVs will likely be near humans, buildings, or sensitive infrastructure, they require secure command authentication.

1.3.3 Short Range UAVs

Short-range UAVs are typically operated near their ground control station or human operator. Many of the missions which are carried out within limited operational areas are based upon photography, inspection, training, local monitoring, small-area mapping. Short-range UAVs usually rely on direct wireless communication connections [21]. As for communication, short-range UAVs usually have a simpler communication architecture compared to larger UAV systems. Since the communication is wireless the channel may be exposed to a variety of attacks. This is of special interest when operating in urban, public, or crowded environments with various wireless devices and potentially attackers that may be nearby. Intelligence studies on UAVs demonstrate that UAV systems are still susceptible to sensor-based and communication-based attacks [22].

1.3.4 Medium Range UAVs

Medium-range UAVs operate over larger mission domains and are generally flown when the UAV must move outside the immediate area of the ground operator. UAVs of this kind may be useful for agricultural surveillance, search-and-rescue, border observation, infrastructure inspections or industrial monitoring. Medium-range UAVs also require stronger and more reliable communication compared to short-range UAVs due to the ability of the aircraft to soar in changing terrain, at larger distances, or in some instances without direct line-of-sight (LOS) communication. It is for this reason that medium-range UAVs can employ stronger antennas, relay nodes, cellular communication, or adaptive link management strategies. Their communication systems must also deal with temporary signal loss, weak coverage, handovers between networks, and shifting environmental conditions [21].

1.3.5 Long Range UAVs

Long-range UAVs are deployed a long distance away from the ground control station and can be used in military reconnaissance, maritime patrol, environmental monitoring, border surveillance, and beyond-visual-line-of-sight (BVLOS) missions. They can fly over long distances and thus need to communicate through satellite links, cellular networks, multi-hop relays, or heterogeneous multi-link communication systems. Long-range missions are generally more complex, costly, and mission-critical than their close-range counterparts. Therefore, the long-range communication requirement for the UAVs is the most demanding. As the distance of UAVs from the operator increases, communication delays and temporary link loss become more serious issues [22], [40].

1.4 UAV Network Topologies

The communication for UAV includes more than aircraft type or mission payload, however, the topology that encapsulates the aircraft network. The communication patterns between these entities are defined by the network topology. The UAV interacts with the ground station, interacts with other UAVs, performs data forwarding in a mesh structure, or communicates using heterogeneous. Topology is extremely important to UAV networks because they are mobile and wireless, and they come under unstable outdoor situations or hostile environments. The UAV networks are usually dynamic unlike the fixed networks. The aircraft may be moving at different speeds, altitudes, going in or out of communications, line of

sight loss, and link swap during a mission. This implies that although the UAV continues to operate, topology changes would probably take place. Most multi-UAV routing research points out that with high mobility and low node density, alongside changes of topology, aerial network design is more difficult than most ground-based wireless networks [25]. Topology matters for security reasons because diverse communication architectures introduce attack vectors.

1.4.1 Single-UAV Communication

The single UAV communication network is the elementary network topology. UAV communicates with ground control station. Such a topology is commonplace in small civilian UAV operations, inspection missions, farming monitoring, photography and mapping. In a single-UAV topology, the primary communication flows are typically command-and-control traffic from the ground station to a UAV, telemetry between the UAV and the operator, and payload data. This architecture is simpler to manage than multi-UAV networks since there is a smaller number of nodes, less routing options, and fewer trust relationships. [26]. This topology is easy, but it does not guarantee to be safe. The wireless connection between the UAV and the ground control station can still be vulnerable. In a single-UAV mission, the ground station is generally the central trusted entity that the UAV relies on to validate communication authenticity and link availability.

1.4.2 UAV Swarms

1.4.2.1 Centralized

An unmanned aerial vehicle (UAV) swarm is a combination of multiple drones working together toward a strategic and mutually cooperating objective of their operation. The mission isn't single aircraft-driven, but multiple UAVs. Swarms are available for the performance of area surveillance, search and rescue, location-environmental monitoring, and military reconnaissance. The primary benefit of swarm's mission is that the mission can receive parallel operation, broad coverage, redundancy, and collaborative actions. In swarm topology, UAVs use a centralized architecture in which the ground controller stations and/or a leader UAV all work together. In certain swarms the mission will be coordinated by a centralized control station or master UAV.

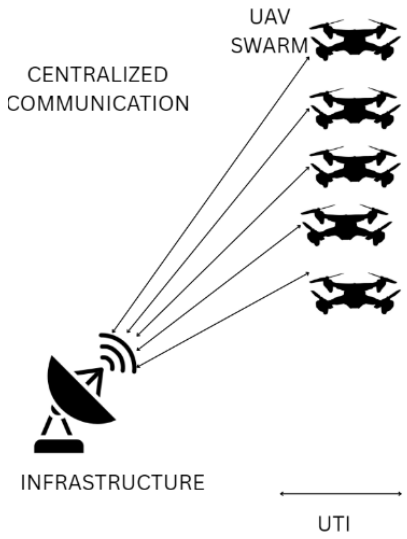


Fig 1. Centralized Communication Topology

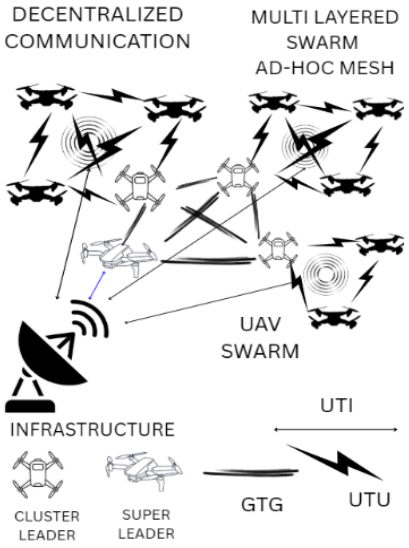


Fig 2. Decentralized Communication Mesh Topology

1.4.2.2 Decentralized

In other swarms the architecture is decentralized so that each UAV decides upon a basis upon direct information received by other nearby UAVs. Recent research on UAV swarm resilience defines swarm networks to be self-organised systems of co-operating, interpenetrating and adapting UAVs systems to varying network conditions [27]. The communication question in UAV swarms is that synchrony relies on accurate sharing of information at the right time. Each UAV is likely to need to provide its starting position, velocity, mission state, sensor readings, route updates, task assignments, and more. If this information is delayed, falsified, or replayed, the swarm may lose forming, duplicate tasks, collide, or make incorrect mission decisions. In swarms, trust distribution can improve resilience against single-node failure, but it does not automatically provide more security than a single-UAV topology. Swarms also introduce a larger attack surface, additional inter-UAV links, and risks such as malicious node injection, Sybil-style impersonation, and compromised members propagating false data.

1.5 Security Challenges and Incidents

Growing utilization of UAVs has created new operational opportunities and poses serious security and safety challenges despite rapid growth in use. UAVs depend on wireless communication, navigation signals, embedded software, remote-control interfaces and sensor payloads. These parts enable UAVs to be flexible but also subject to cyber threats. Unlike common information systems, a compromised UAV can cause both physical and digital effects. If an attacker intercepts data, injects commands, spoofs navigation, jams communication or takes control of the UAV, the consequence could be mission failure, a loss of sensitive information or physical damage, disruption of public services or risk to human lives. The biggest security challenges related to UAV communications are eavesdropping, command injection, spoofing, jamming, replay, man-in-the-middle attacks, malware, unauthorized access, and physical capture. These threats are particular interest as the UAVs are typically sent through open wireless channels and can be used as agents to monitor, jam or attack the communication link. Furthermore, UAVs are also typically limited by battery, payload, processing capacity, and bandwidth. Thus, the security mechanisms should be sufficient to protect the mission but as light in weight as possible when flying resource-constrained airborne platforms. These actual UAV incidents illustrate that these issues are not hypothetical. Drone-related incidents have hit airports, military operations, critical infrastructure, politicians, and even active war sites.

1.5.1 RQ-170 Sentinel

The 2011 RQ-170 Sentinel incident was one of the most severe cases where a high-value military UAV was lost in adversarial territory. In December 2011, Iran revealed that U.S. RQ-170 UAV had been captured [30]. Signal interference, GPS spoofing, and command-and-control disruption were mentioned by some, some other officials were inconclusive with the details in doubt, that the aircraft might have been the victim of technical failure or loss of control [31]. The significance of this attack extends beyond the question of whether one specific spoofing allegation was proper. More importantly, it demonstrated that UAV loss can create strategic consequences. A captured UAV could contain sensitive sensor technology, mission data, communication systems, airframe design, or operational methods. If an adversary recovers the aircraft, this allows for reverse engineering, misappropriation of intelligence and propaganda value to occur. For UAV communication research, the RQ-170 case emphasizes the role of resilient navigation, authenticated command links, anti-spoofing, fail-safe behaviour.

1.5.2 UAV in the Russian-Ukraine War

The war between Russia and Ukraine has resulted in one of the largest and recent demonstrations of UAVs in the modern war system. UAVs have supported reconnaissance, artillery spotting, strike operations, electronic warfare support and battlefield awareness on both sides. The war has demonstrated that UAVs that are commercially available and cost-effective can be rapidly used to accomplish military ends. Small UAVs are employed not only as airborne cameras, but are also weaponized as expendable attack platforms, loitering munitions, and tactical information systems. One of the major lessons of this war, is the importance of electronic warfare. According to reports there are extensive use of jamming and spoofing, while some systems have implemented electronic control techniques such as Fiber-optic control to avoid radio-frequency jamming [32] [33]. This shows that the communication link is the most vulnerable one to be targeted in the UAV system. The Russia-Ukraine conflict further demonstrates the operational importance of resilient UAV communication systems under contested electromagnetic conditions. In 2025 some reports have described efforts to enhance the resistance of drones towards jamming and spoofing with advanced antenna technologies [34]. This validates that UAV communication protection cannot depend solely on a static defence. It needs to handle resilience and fall-back procedures when a channel becomes unavailable or untrusted.

Chapter 2

Research Context and Objectives

2.1 Problem Statement

As their operational value has grown, that brings with it a growing security risk and not just in relation to the fact that they require open wireless communication to remain functional. Therefore, a cyberattack on a UAV can inflict more than just a data breach. Such behaviour can also cause mission failure, loss of control, transmission of sensitive data, and harm to the equipment itself. This thesis concentrates on the issue of how to protect UAV communication systems in adversarial environments in which wireless connections may be unreliable, jammed, spoofed, intercepted, or manipulated. The latest UAV security survey finds that spoofing, false-data injection, jamming, confidentiality threats, integrity attacks, and availability attacks are among the main security threats against UAV systems [36]. Other studies on UAV security also emphasized that UAVs have many attack surfaces, such as an aircraft node, ground control station, wireless communication links, payload systems, and Internet-connected infrastructure [37]. This large attack surface makes UAV security particularly complex. This is compounded in UAV deployments, where resource constraints are common. Consequently, the security mechanisms in strong ground-based systems are not necessarily directly suited to UAVs and security requirements such as computational cost, communication overhead, power and real-time response. As a result, UAV security will need a delicate trade-off between security, performance, power usage and task need.

2.2 Motivation

The rationale of this research lies in the fact that UAV operations are increasingly reliant on secure communication. So, if communication is exposed, the whole mission will be impacted. This is a critical issue at present given the growing use of UAVs in the civil, commercial, emergency, and military sectors. In all these instances, the UAV

must secure the data that it transmits. False command, tampered telemetry message, or captured payload stream can cause wrong decisions or failure of mission. A key reason is the fact that UAVs use wireless communications links, which are inherently vulnerable. Depending on the mission, UAVs might exchange through radiofrequency (RF), Wi-Fi, cellular or satellite links from one UAV or another. Each link would have its own operational advantages, but each one also carries a different security danger. RF links may also get jammed, Wi-Fi links can be abused through wireless exploitation techniques, cellular links can rely on public infrastructure, and satellite connections can have delay and availability constraints. Motivation is further heightened by the reality that UAVs frequently do much in adversarial or degraded environments. Consequently, the UAV should be not only able to protect communication during the usual conditions but also be able to safely manoeuvre in the event of communication disruption, delay, or partial unavailability. Real world incidents demonstrate that the dangers are real, not just theoretical and triggers more research into UAV communications mechanisms that can provide stronger security solutions.

2.3 Literature Review

Secure communication is one of the most important requirements for UAV systems. Unlike wired or safe networks, UAV communication happens over open radio channels that can be intercepted, jammed, spoofed, replayed or tampered with by an adversary. Thus, since secure communication security is increasingly the focus, many authors have emphasized protecting the confidentiality but also integrity, availability, and resistance under extreme conditions. Studies on cybersecurity have shown that UAV systems face various communication-layer and cyber-physical threats. Tsao et al. reviewed issues related to UAV security privacy as well as threats to UAV communications, control, Internet-connected drone infrastructure [37]. They find in their work that UAV security is not just a matter of protecting the aircraft, the ground control station, communication link, data infrastructure and user access mechanisms – all serve as attack surfaces. Likewise, Shafique et al. survey security protocols and security holes in unmanned aerial systems are also given a glimpse of attacks against communication and control systems [36]. Other studies have been more dedicated to UAV communication and cryptology protection. Aissaoui et al. offer a review of cryptographic techniques to secure communication on UAVs and highlight that UAV missions need robust security and safety guarantees, particularly when flying within a public airspace [38]. One frequent theme in the literature is to worry about the vulnerability of wireless links from UAV wireless links. These connections are vulnerable to attack. Such links are susceptible to attacks. It is important to note that UAV communication links, as Patel and Cherukuri found, depend on wireless communication threat exposure such as eavesdropping, data interception and signal jamming [39]. All of this strengthens the need for communication protocols able to stay secure under the pressure of link quality change or attack on the transmission. Lightweight cryptography is a promising avenue of research for the time being. Classic cryptographic methods can grant strong security, but they could come with some type of computational and energy overhead as well as latency and/or implementation complexity. For this reason, researchers focused on lightweight encryption, authenticating encryption, lightweight authentication mechanisms, efficient key management techniques for UAV communication systems. Sarkar et al. present a systematic survey for safe communication in drone network security, with special attention to lightweight encryption, secure key management, and post-quantum cryptographic direction for resource-constrained UAV systems [40]. They demonstrate that choice of security mechanisms should depend on the layer of communication, mission type, and available resources. UAV-oriented works have also made recent evaluations on Ascon and other lightweight cryptographic algorithms for the UAV

communication. Patel and Cherukuri present a lightweight cryptography algorithms evaluation of UAV networks and argue that Ascon is a practical tool in UAV, and its efficiency also makes it a reasonable choice when the computation capacity and even the energy capacity of a UAV system is limited. Their research evaluates lightweight cryptographic choices for UAV communication and identifies Ascon-128a as an attractive candidate for trade-offs between efficiency and security [39] [41]. However, research shows that encryption alone is not sufficient for secure UAV communication. He et al. proposes a security-enhanced lightweight authenticated key-agreement protocol for UAV communication and argue that existing key-agreement systems may be vulnerable when temporary session information is compromised [42].

2.4 Research Gap

Despite these contributions, there are still important gaps in the current research. Most studies concentrate on particular security mechanisms, like encryption, authentication, intrusion detection, spoofing detection, or jamming detection. While important, these mechanisms are frequently considered independently of the others and do not always address how they should operate together within a complete UAV communication architecture. Some of the existing works are broad surveys of UAV cybersecurity threats and the main risks against UAV systems but do not always propose a unified protocol design capable of secure communication. That leads to the requirement for an approach. Therefore, the reviewed literature shows a clear direction for this thesis. Secure UAV communication requires more than choosing a cryptographic algorithm.

2.5 Research Aim

This research proposes a mission-oriented protocol for securing UAV communication in adversarial environments. The protocol proposed by this work aims to mitigate the security weaknesses of current communication approaches by bringing mission-scoped identity, mutual authentication, lightweight cryptographic key establishment, replay protection, and security capabilities into a unified protocol structure. Its goal is not only to encrypt communication between UAVs and the Ground Control Station, but also to provide a full mission-aware security model. The UAV, Ground Control Station, authorized operators, mission identity, communication channels, and available links are all considered within one mission context. This enables cryptographic material, authorization rules, trust decisions, and communication policies to remain bound to the mission purpose instead of being reused across different missions. The research focus is primarily on securing the authentication and key-establishment process between the UAV and the Ground Control Station. This procedure enables both entities to verify each other's identity, agree on fresh session key material, bind the exchange to the current mission, and defend against multiple attacks. Because UAVs operate over open wireless channels, the authentication phase assumes that an adversary can observe, modify, inject, delay, or replay messages. Thus, the goal is to provide strong cryptographic assurance while remaining suitable for resource-constrained UAV platforms. The other goal is to support secure and structured multi-link communication. UAVs may use RF, Wi-Fi, LTE/5G, satellite, or UAV-to-UAV relay links during a mission. These communication links vary in latency, bandwidth, availability, and susceptibility to attack, which is especially important in contested environments where one link may be jammed, degraded, or disconnected.

Chapter 3

Background on UAV systems

3.1 Software Architecture

A UAV's software architecture is the logical control structure used by the aircraft that enables it to sense the environment around it, estimate its state, execute flight-control decisions, as well as communicate with external systems to accomplish mission objectives. Software consists of embedded flight-control firmware, sensor drivers, algorithms, navigation modules, protocols, mission-management logic, payload-control software, ground-control software, and sometimes companion-computer applications for higher-level autonomy. The software architecture is represented as a layered system. However, the structure may be different for each UAV depending on the type of UAV. The device driver layer is the lowest level and is designed for the interaction between the UAV software and physical hardware. This layer consists of the drivers for GPS receivers, sensors, barometers, servos, motors, and other onboard devices. The purpose of this layer is to allow the upper software components to get hardware measurements and control actuator commands even though the hardware details are not managed directly by this layer. This is critical in the architecture due to the flight-control system relying on precise as well as timely sensor information. The operating system layer is located above the driver layer. UAVs typically use real-time or embedded operating systems such as NuttX or Linux. The operating system manages tasks, scheduling, memory, hardware resources, timing, and process execution. The middleware layer is above the operating system and includes communication and coordination services between all the different software modules. This layer also contains components such as CSP and message passing. Middleware enables various parts of the UAV software to share information with other parts of the same platform without being tightly connected to each other. Such design enhances modularity, enabling individual software components to be developed, tested, or replaced without rewriting the entire system. The application layer is at the top and represents the functional software that underpins the main UAV functions. This layer consists of modules like control, position, attitude, motor, sensor, navigation, and communication. These are the functions which handle the behaviour of the specific UAV. [43]

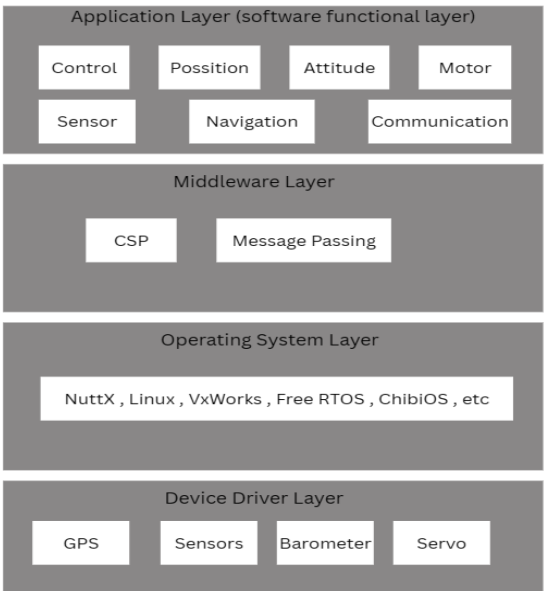


Fig. 3 Software Architecture

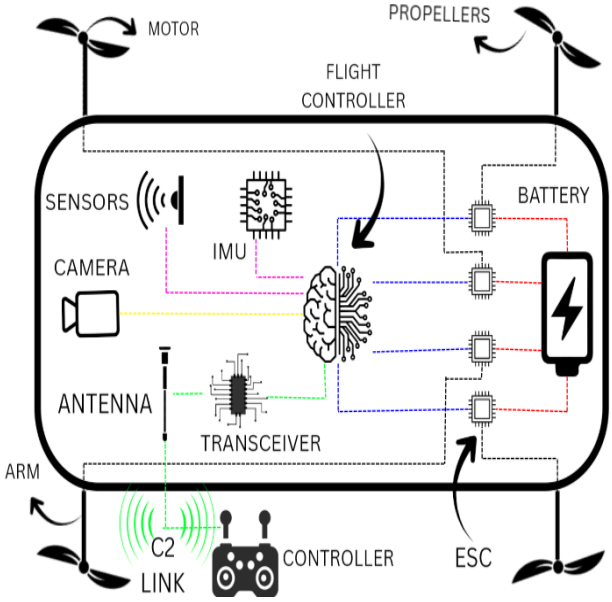


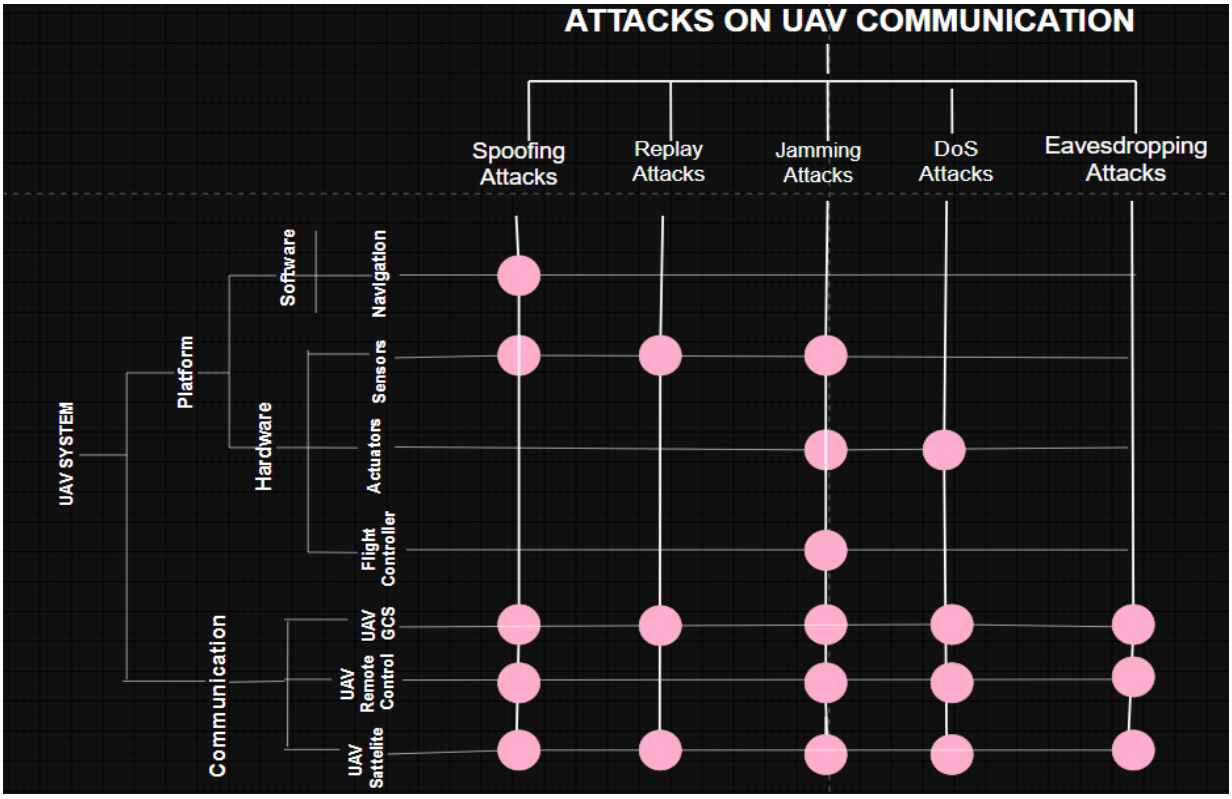
Fig. 4 Hardware Architecture

3.2 Hardware Architecture

The common architecture of a UAV hardware comprises a variety of related parts facilitating the sense of the environment, information reception, communication with the user and moving objects generated by the flight. As indicated in the figure, a primary part is called flight controller, in which data is transmitted from onboard sensors and this data is input into the propulsion system for control purposes. The flight controller of multiple UAVs consists of integration or integration with the inertial measurement unit (IMU), GPS, barometer, compass, and additional sensing systems. The sensor subsystem delivers the data necessary for stabilization, navigation, and mission execution. The IMU measures acceleration and angular motion, and GPS provides positioning information. Depending on the mission, other sensors including barometers, cameras, rangefinders, or optical-flow sensors may be included or added. The flight controller then processes this information to estimate the state of the UAV and determine how the aircraft should respond. The propulsion subsystem consists of the battery, electronic speed controllers (ESC) motors, and propellers. The battery provides power to the UAV and the ESCs regulate the power delivered to each motor. The motors then rotate the propellers to produce lift and motion. In this communication subsystem are the antenna, transceiver, and command-and-control link between the UAV and ground controller. This enables sending commands and receiving telemetry from the operator. Depending on the type of UAV, this link may use different wireless technologies. The communication subsystem is also critical to security because it houses control commands, telemetry, mission updates and sometimes payload information. The mission is responsible for the payload subsystem. In this figure is the example payload as a camera. Thermal cameras, LiDAR sensors, radar, communication relays, mapping equipment or delivery mechanisms may all be carried in other UAVs. Larger UAVs might also include a companion computer which processes payload data, executes computer vision algorithms, or provides autonomous decision support. That is, the figure illustrates common UAV hardware structure but is not the same for every UAV. Small commercial UAVs might have small integrated boards, while large UAVs might have complex avionics.

3.3 Security Threats in UAV

As shown in Figure, UAV communication threats can appear across several system components and links. The attack matrix highlights five major threats showing that UAV communication risk is distributed across multiple channels. This section therefore focuses on these attacks.



3.3.1 Spoofing Attacks

Spoofing is an attack in which the adversary impersonates a legitimate communication source. When the attacker tries to communicate with the UAV, they may impersonate the Ground Control Station, remote-control transmitter, or another UAV. The goal is to trick the UAV to receive a message as if it was from a trusted entity. In the attack matrix, spoofing shows up in several communication components, indicating that every external UAV communication interface can end up being a spoofing target. Because UAV commands can cause physical effects, this attack is risky. They could make a UAV change route, alter altitude, return home or land, activate a payload, abort a mission, or send sensitive data to an attacker. Spoofing grows more complicated in UAV swarms and mesh networks. That is, a malicious node may impersonate a trusted UAV, which would send false coordination messages, routing information, or status updates. Due to dynamic change of UAV networks in flight, it cannot be done using only static network [28], [29]. To guard against spoofing, the proposed protocol necessitates mission-scoped database and mutual authentication among endpoints. Before accepting a command or mission update, it verifies that the sender is an authorized entity for the current mission.

3.3.2 Replay Attacks

Replay attack happens when an adversary records a valid UAV message and retransmits it later to create an unauthorized effect. There is no need for the attacker to decrypt or modify the message. Instead, the attacker reuses a legitimate message from an earlier time. Replay attacks appear across communication related components in the given attack matrix. This is because

any wireless UAV link can be monitored and recorded by an adversary. Replayed coordination messages may disrupt formation control, routing, or task assignment as part of swarm or mesh scenarios. Replay is identified as a serious communication threat in UAV security literature as UAV messages need to be both authentic and fresh to be trusted [36], [37]. For this thesis, replay attacks motivate the use of nonces, timestamps, counters, mission epochs, and replay windows.

3.3.3 Jamming Attacks

Jamming is an availability attack in which an adversary intentionally interferes with UAV wireless communication or navigation signals. Its goal is to prevent the UAV and its legitimate communication peer from exchanging useful messages. Jamming is one of the most practical communication attacks against UAVs, since most UAVs rely on wireless links. The attack matrix shows jamming against communication-related components. These signals cause disruption to short-range and long-range UAV communication. If the control link is jammed, the UAV may cease receiving direct commands. Availability is the focus of the jamming attack. UAV security studies indicate that jamming is a significant threat to communication activities, due to interruptions to control, telemetry, and mission data exchange [36], [37]. In practice, the Russia–Ukraine war also demonstrates that jamming is an important electronic-warfare method for disrupting operations performed by drones and weakening the operational effectiveness of an unmanned aerial vehicle [33], [35]. Jamming underlies the necessity of secure multi-link resilience for this thesis. The protocol must not rely on a single fragile wireless channel. It must support link specific keys, link switching and safe-mode behaviour in case trusted communication cannot be restored.

3.3.4 Denial of Service (DoS) Attacks

A denial-of-service (DoS) attack attempts to make UAV communication services unavailable or unreliable. While jamming is usually a physical-layer denial-of-service attack, DoS can also occur at higher communication layers. An adversary may flood the endpoint with fake packets to overload communication, exhaust memory or force repeated expensive cryptographic operations. DoS attacks are especially dangerous for UAVs because UAVs are resource constrained. An attacker does not always need to fully break the system. Security surveys identify DoS as a major availability threat because it can disrupt communication and prevent safe operation [67], [68]. For this thesis, DoS attacks motivate an asymmetric and resource-aware protocol design. The Ground Control Station can manage more state, retries, and mission lifecycle control, while the UAV should maintain only minimal temporary state before authentication is completed. The UAV should use TTL-bounded temporary state and LRU-bounded caches. This reduces the risk that an attacker can exhaust UAV memory or processing capacity by flooding it with fake requests. DoS attacks also motivate degraded-mode behaviour. If the UAV loses communication because of DoS, it should not blindly accept unsafe commands or continue indefinitely without trust. Instead, it should

attempt authenticated fallback links, use safe mission policies, and eventually enter safe mode if communication cannot be restored within the mission-defined timeout.

3.3.5 Eavesdropping Attacks

Eavesdropping is a passive communication attack in which an adversary listens to UAV traffic without modifying it. Since UAV communication is wireless, an attacker may capture exchanged messages. Although eavesdropping does not directly change UAV behaviour, it can still seriously compromise a mission. Intercepted payload data may reveal surveillance video or military intelligence. UAV cybersecurity surveys identify eavesdropping and confidentiality loss as major threats against UAV communication systems [36], [37]. Eavesdropping is especially serious in military, law-enforcement, emergency-response, and critical-infrastructure missions. In those cases, the attacker may not need to control the UAV, simply learning what the UAV sees, where it flies, or when it communicates may be enough to compromise the mission. For this thesis, eavesdropping motivates the need for confidentiality and key separation. A secure authenticated key exchange is required to establish fresh session keys between the UAV and GCS. However, using only one general key for all mission traffic is not ideal. Instead, independent keys are derived. This limits the damage if one key or channel is exposed and supports stronger mission compartmentalization.

Chapter 4

Contribution

Threat Model

This thesis considers a UAV mission environment in which communication takes place over potentially hostile and unreliable wireless channels. The adversary is an active network attacker with Dolev–Yao-style control over the communication medium. Under this model, the adversary can observe, intercepting, delaying, dropping, replaying, modifying and injecting messages transmitted over RF, Wi-Fi, LTE/5G, satellite, and UAV-to-UAV links. Although the adversary is assumed to have strong communication capabilities, it is assumed to be computationally bounded in the cryptographic sense. Specifically, the adversary is a probabilistic polynomial-time attacker and is not assumed to be capable of breaking standard cryptographic assumptions. The adversary may compromise communication links either individually or simultaneously. Consequently, the proposed protocol does not assume that any single wireless channel is permanently available, confidential, or authentic before cryptographic verification is performed. The protocol does not claim to fully eliminate organizational insider abuse. The protocol assumes that long-term private keys are protected by the Trusted Platform Module (TPM), while volatile session keys and sensitive mission data are zeroized when Safe Mode, mission termination, or tamper response is triggered.
Under this threat model, the main security goals of the proposed protocol are to preserve the confidentiality of mission traffic, ensure the integrity and authenticity of commands and telemetry, provide replay protection, resist impersonation, establish fresh session keys securely, separate traffic keys across directions, channels, and links, contain compromised UAVs, maintain safe behaviour under link failure or jamming, and securely erase sensitive material at mission termination or during emergency response.

4.1 Phase A: Pre-Deployment Provisioning

Phase A constitutes the foundational trust establishment stage. All operations are performed within a physically secure environment prior to deployment. The principal objective is to generate, validate, and cross register the long-term cryptographic identities upon which every subsequent protocol phase depends.

A mission-scoped database serves as the authoritative root of trust for the entire mission and air-to-air mesh communication. It is maintained exclusively by the Ground Control Station (GCS) and contains the registered pseudonymous identities of all participating entities, including the GCS, all UAVs assigned to the mission, and all authorized operators. For UAV identities, the database additionally maintains an associated role and status flag indicating their current trust state.

At the beginning of the mission, each registered UAV identity is marked as Trusted, signifying that its authenticity has been successfully verified and that it is permitted to participate in protocol operations. During mission execution, a UAV identity may transition from Trusted to Quarantine or Revoked state. A state transition is permitted only if triggered either by the machine-learning-based anomaly detection system or by a Trusted Platform Module (TPM) tamper. Following revocation, the UAV is considered untrusted and is excluded from further trusted participation for the remainder of the mission.

Each UAV stores a synchronized local read-only replica of the mission database for runtime authentication and offline operations. Enforcing read-only access ensures that the compromise of an individual UAV cannot propagate unauthorized trust modifications across the global trust state. Whenever the authoritative database is modified, an updated replica is transmitted over the authenticated CTRL channel. The UAV accepts and installs the update only after verifying the accompanying digital signature using the provisioned GCS public identity.

When an individual UAV loses connectivity with the GCS, it queries the other swarm members for the most recent signed replica of the mission database. This enables recovery of the latest trust state information despite temporary loss of direct GCS communication. If communication with the GCS is lost by the entire swarm, the predefined link-loss timeout is activated, upon which the UAVs autonomously enter Safe Mode.

To mitigate the inherent limitation that human operator may be unable to identify abnormal behaviour during flight and promptly transition the corresponding identity state in real time, the protocol incorporates a machine-learning-based anomaly detection component operating on the GCS side.

The anomaly detection component continuously evaluates UAV telemetry using an LSTM autoencoder trained on nominal flight behaviour. Reconstruction error is measured over sliding telemetry windows and compared against an adaptive threshold to generate an anomaly score, which the protocol interprets as a per-identity risk score. To translate this score into security actions, the protocol employs three-level mechanism.

Because anomaly scores directly influence Quarantine and Revoked states, the protocol also treats the LSTM component as security sensitive. An adversary may attempt evasion by crafting telemetry that remains below the suspicion threshold or poisoning by causing legitimate telemetry to appear anomalous. Therefore, anomaly-based decisions should be corroborated with sustained Grace Window evidence, cryptographic identity checks, and operator or maintenance review before irreversible revocation is finalized.

THRESHOLD	DESCRIPTION	SECURITY RATIONALE
Suspicion	Anomaly score exceeds expected behaviour baseline	UAV enters Quarantine state

Critical	Suspicion * (1 + δ) (δ configurable factor)	State is changed only if the score remains for the full grace window duration
Grace Window	Sustained duration before transforming the UAV state	Prevents false positive revocations from transient anomalies

Table 1. Machine Learning Table

Quarantine State: When a UAV enters the Quarantine state, it is allowed to continue executing its assigned mission tasks. However, its network privileges and operational influence are restricted. The UAV is excluded from sensitive swarm operations, until its trust status is restored by the protocol. The UAV exits Quarantine and returns to a Trusted state if the anomaly score remains below suspicion threshold for the entire Grace Window. This recovery mechanism ensures that trust is restored only after normal behaviour has been observed, preventing rapid state oscillations.

Revoked State: When a UAV's anomaly score remains above the Critical Threshold for the entire Grace Window, the UAV remains quarantined and a rekey operation is initiated. Rekeying is used to contain possible session-level compromise and to monitor whether the anomalous behaviour subsides. If the anomaly score subsequently drops below the Suspicion Threshold, the UAV remains subject to the Quarantine state. Otherwise, if anomalous behaviour persists, the UAV identity is transitioned to Revoked state and is excluded from trusted participation for the remainder of the mission. If revocation is caused by a TPM tamper event, the UAV shall be regarded as potentially physically compromised and must undergo physical inspection and integrity revalidation before re-enrolment in a future mission.

A TPM tamper event is not treated as a trustworthy self-report if it is delivered only over a potentially compromised mission link. The GCS accepts tamper-based revocation only when the evidence is received through an independent trusted maintenance or out-of-band channel, or through verifiable signed TPM attestation when available. If such a trusted notification path is unavailable, the GCS treats the UAV as suspicious, relies on the anomaly-detection and communication evidence, and requires physical inspection before any future re-enrolment.

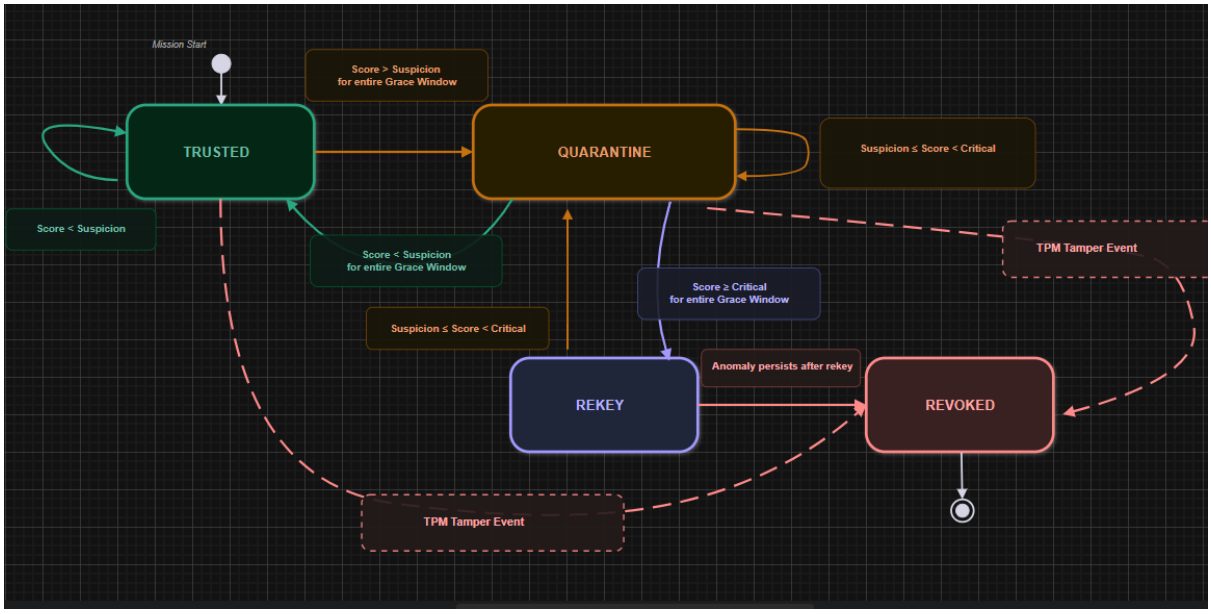


Fig. 5 Anomaly Score Conversion

Each participant X generates a per-mission long-term identity used exclusively for authentication.

$S_X \leftarrow RNG_{256} \oplus Jitter_{256}$
$SK_X \leftarrow CSPRNG\ (S_X)$
$PK_X \leftarrow SK_X * B_{ED25519}$
$ID_X \leftarrow$ Serial Number
$PID_X = H\ (PK_X \parallel MID \parallel ID_X)$

SYMBOL	DESCRIPTION
S_X (256-bit)	Entropy seed combined with exclusive or jitter
SK_X (256-bit)	Private key generated using the seed
$B_{ED25519}$	Base point of Ed25519 elliptic curve
PK_X (256-bit)	Public key derived from private key using elliptic curve scalar multiplication
PID_X (256-bit)	Pseudonymous Identity
H	Hash function

Table 2: Cryptographic Symbols used in Identity Provisioning

Following key generation, each pseudonymous identity is enrolled into the mission trust database through a dedicated enrolment tool that interacts with the database via an authenticated management API.

Local Bundle $\leftarrow$ (PK_x, PID_X) = 512-bit Bundle
Database Record $\leftarrow$ (PID_X, PK_X, Status $_X$, Role $_X$)

The Mission Policy Package is created by a dedicated script and transferred to both UAV and GCS through a secure out-of-band (OOB) channel. This package encapsulates the static mission parameters along with golden tokens and safe mode policy that govern protocol behaviour. Externalizing these parameters enables the operator to tailor the protocol to the specific operational environment, and security aggressiveness in accordance with the assessed threat level.

PARAMETER	DESCRIPTION	SECURITY RATIONALE
MID (32-bit)	Unique mission instance identifier	Mission Identifier (Predefined Generated by RNG)
MR (8-bit)	Maximum handshake re-attempts	Prevents infinite retries from GCS to UAV
TRF (16-bit)	Response wait time before timeout	Tuned on RF propagation
TWi-Fi (16-bit)	Response wait time before timeout	Tuned on Wi-Fi interference and congestion
TLTE (16-bit)	Response wait time before timeout	Tuned on LTE
TSAT (16-bit)	Response wait time before timeout	Tuned for SAT propagation and processing overhead
TD (16-bit)	Acceptable timestamp tolerance	Replay protection window
WS (16-bit)	Window size for packet	Sequence tracking
RT (32-bit)	Maximum allowed duration of epoch	Decreases cryptanalysis exposure
RB (32-bit)	Maximum byte volume of epoch	Maintains AEAD bounds
GF (8-bit)	Grace factor	Old-new key overlap window
LL (16-bit)	Duration of total communication failure before Safe Mode	Enables UAV autonomous safe mode policy
FHSSCH	Hop Channels	Expand hopping diversity
FHSSTRT	Start Frequency	RF spectrum
FHSSEND	End Frequency	RF spectrum

Table 3: Static Mission Parameters

Mission Policy Package also defines the UAV's **autonomous response**, which is triggered due to sustained total communication loss or a TPM tamper event. Upon activation, the UAV immediately executes zeroization and recovery sequence. This sequence is designed to first destroy sensitive cryptographic material, mission data and then transition the platform into a controlled recovery.

ACTION	DESCRIPTION	SECURITY RATIONALE
Zeroize Keys (8-bitmask)	Destroys all volatile cryptographic material	Prevents recovery of mission keys
Erase Mission Data (16-bitmask)	Erase sensitive data stored on UAV	Protects mission confidentiality
Flight Action (8-bit Enum)	Executes Return to Home (RTH) if navigation is available otherwise Emergency Land	Ensures UAV recovery and safe handling by responders

Table 3: Safe Mode Policy Sequence Actions

The Golden Token is a cryptographically authenticated emergency command, digitally signed by the operator, that instructs a designated UAV to immediately transition into Safe Mode and carry out the specified home point recovery behaviour, which is executed by the Flight Controller

PARAMETER	DESCRIPTION
Token ID (256-bit)	Random nonce stored in the GCS used-token list after acceptance
Action (8-bit)	Enter Safe Mode with Home point
PID_{UAV} (256-bit)	Target UAV
Home Point (64-bit)	Coordinates for autonomous landing
Payload (584-bit)	PLD = ("Golden Token" || Token ID || Action || $PID_{UAV,}$ Home point)
Signature (512-bit)	Sign_Ed25519 ($SK_{Operator}$, Payload)

Table 4: Golden Token

The Golden Token is mission-bound because PIDUAV is derived from the UAV public key, mission identifier MID, and serial identifier. Therefore, a captured token from a previous mission cannot be replayed in a later mission with a different MID. Within the same mission, the GCS maintains a used-token list indexed by Token ID and rejects any Golden Token whose Token ID has already been accepted, preventing double use of the same emergency token.

UAV's long-term private key SK_{UAV} is hardware protected within the Trusted Platform Module (TPM), alongside secure counters and integrity state. The local bundle and the replica of mission database are stored in secure local storage external to the TPM.

4.2 Phase B: Authentication Key Exchange (AKE)

The Authentication Key Exchange phase is executed while the UAV remains within Visual Line of Sight (VLOS) of the Ground Control Station (GCS). Despite the physical proximity associated with VLOS operation, the communication environment is conservatively assumed to be adversarial. The protocol therefore establishes mutual authentication between the endpoints and derives fresh master session key, enabling secure continued operation in Beyond Visual Line of Sight (BVLOS) scenarios.

The communication adopts an asymmetric handshake model in which the Ground Control Station (GCS) acts as the handshake initiator, while the UAV operates as a minimal responder. This design reflects the differing capabilities and operational roles of the two endpoints. The GCS, having greater computational resources and direct operator oversight, is responsible for initiating and managing the progression of the exchange, including timeout handling, retransmission control, and session coordination. In contrast, the UAV remains in a passive listening posture and performs only the actions necessary to validate incoming messages, generate the required cryptographic responses, and return to standby if verification fails. This minimizes onboard processing overhead, reduces unnecessary state retention, and

limits the UAV's attack surface, while preserving mutual authentication and secure session establishment.

Rule: Upon receiving a handshake message, the receiving endpoint first performs a local verification procedure before committing computational resources to the exchange. Specifically, it verifies that the claimed peer pseudonymous identity exists and matches an entry in the trusted local database or replica. It then checks that the received nonce has not been previously observed for the corresponding handshake attempt and evaluates timestamp freshness against the configured time-difference tolerance. Because MSG1 is unsigned, Degraded Sync Mode is not entered solely on the basis of an out-of-window MSG1 timestamp. If MSG1 fails timestamp freshness, the UAV discards it unless the mission policy explicitly enables operator-supervised degraded synchronization during VLOS. For signed MSG2 and MSG3, an out-of-window timestamp with a fresh nonce may be recorded as a warning and allowed only for the current handshake attempt; all nonce, PID, and signature checks remain mandatory, and the endpoint exits this state after key confirmation. The endpoint then stores the relevant temporary handshake state in a bounded cache with Time-To-Live (TTL)-based expiry and Least Recently Used (LRU)-based eviction.

The GCS initiates the Key Exchange by generating a fresh ephemeral key pair, nonce and records its current timestamp. These values are assembled into the initial handshake message.

$ES_{GCS} \leftarrow HRNG_{256} \oplus Jitter_{256}$
$ESK_{GCS} \leftarrow CSPRNG\ (ES_{GCS})$
$EPK_{GCS} \leftarrow ESK_{GCS} * B_{X25519}$
$N_{GCS} \leftarrow HRNG_{256}$
$T_{GCS} \leftarrow$ Current Time
$MSG_1 \leftarrow$ {Handshake Initialization, PID_{GCS}, PID_{UAV}, N_{GCS}, T_{GCS}, EPK_{GCS}}
Timer $_{LINK} \leftarrow T_{LINK}$ (On timeout regenerate)

SYMBOL	DESCRIPTION
ES_X (256-bit)	Entropy seed combined with exclusive or jitter
ESK_X (256-bit)	Ephemeral private key generated using the seed
EPK_X (256-bit)	Ephemeral public key derived from private key using elliptic curve scalar multiplication
N_X (256-bit)	Nonce
T_X	Timestamp

Table 5: Cryptographic Symbols used in Authentication Key Exchange

The ephemeral entropy seed is 256 bits for both endpoints. This concrete requirement is separate from the symbolic Tamarin model, where ephemeral secrets are represented as fresh, unguessable values; the symbolic model therefore does not replace the need for 256-bit entropy in the implementation.

UAV Application of the Rule. The UAV applies the above rule upon receiving MSG_1. Only if all local verification checks succeed does the UAV generate its own ephemeral key, thereby avoiding unnecessary computation and resource consumption. If any of the verification checks fail, the UAV silently discards the packet without generating a response.

$ES_{UAV} \leftarrow HRNG_{256} \oplus Jitter_{256}$
$ESK_{UAV} \leftarrow CSPRNG\ (ES_{UAV})$
$EPK_{UAV} \leftarrow ESK_{UAV} * B_{X25519}$
$N_{UAV} \leftarrow HRNG_{256}$
$T_{UAV} \leftarrow$ Current Time
Payload $_{UAV} \leftarrow$ (UAV confirm || PID_{UAV} || PID_{GCS} || MID || EPK_{UAV} || EPK_{GCS} || N_{UAV} || N_{GCS} || T_{UAV} || T_{GCS})
Sign $_{UAV}$ = Ed25519_Sign (SK_{UAV}, Payload $_{UAV}$)

MSG_2 ← {Handshake Response, PID_{UAV}, PID_{GCS}, N_{UAV}, T_{UAV}, EPK_{UAV}, Sign $_{UAV}$}
Timer $_{LINK}$ ← T $_{LINK}$ (On timeout discard MSG_3)

GCS Application of the Rule. The GCS applies the above rule upon receiving MSG_2. Only if all local verification checks succeed and the received UAV payload signature is successfully verified does the GCS construct its own confirmation payload and generate the corresponding signature. If any local verification check fails, or if signature verification is unsuccessful, the GCS discards the message and treats the handshake attempt as invalid and restarts from the beginning until it exceeds the configured max retries.

Ed25519_Verify (PK_{UAV} || Payload $_{UAV}$ || Sign $_{UAV}$)
Payload $_{GCS}$ ← (GCS confirm || PID_{GCS} || PID_{UAV} || MID || EPK_{GCS} || EPK_{UAV} || N_{GCS} || N_{UAV} || T_{GCS} || T_{UAV})
Sign $_{GCS}$ = Ed25519_Sign (SK_{GCS}, Payload $_{GCS}$)
MSG_3 = {Handshake Verification, PID_{GCS}, PID_{UAV}, EPK_{GCS}, EPK_{UAV}, N_{GCS}, N_{UAV}, T_{GCS}, T_{UAV}, Sign $_{GCS}$}

UAV Application of the Rule. The UAV applies the above rule upon receiving MSG3. Only if all local verification checks succeed and the received GCS payload signature is successfully verified, it proceeds to compute the shared secret. If any local verification check fails, if signature verification is unsuccessful, or if a timeout occurs, the UAV silently discards the message and returns to passive listening posture. This process guarantees that both endpoints will compute the same shared secret.

Ed25519_Verify (PK_{GCS} || Payload $_{GCS}$ || Sign $_{GCS}$)

The signature binds both ephemeral public keys, both nonces, both timestamps, and a fixed protocol label to the authenticated transcript. This construction prevents man-in-the-middle substitution, cross-protocol confusion, and reflection attacks. Any tampering with the exchanged handshake parameters is detected by the verifying endpoint, since both the GCS and the UAV independently reconstruct the expected peer payload and will observe a mismatch during signature verification.
Both endpoints independently compute the Diffie-Hellman shared secret, the transcript, and the salt used to create the Session Master Key (SMK).

DHGCS = X25519 (ESKGCS, EPKUAV)
DHUAV = X25519 (ESKUAV, EPKGCS)
DHGCS = DHUAV
TRSCRPT ← H (TLV{MSG_1} || TLV{MSG_2} || TLV{MSG_3})
SLT_X ← H (PID_{GCS} || PID_{UAV} || TRSCRPT)
PRKX HKDF Extract (SLTX, DHX)
SMK_X ← HKDF $_{Expand}$ (PRK_X, “Session Master Key” || MID, L = 256-bit)

SYMBOL	DESCRIPTION
DHX (256-bit)	Diffie-Hellman shared secret
TRSCRPT	Transcript
SLT_X	Salt
PRK_X	HKDF result
SMK_X (256-bit)	Session Master Key

Table 6: Cryptographic Symbols for Session Master Key computation

After successful computation of the shared secret, both the GCS and the UAV securely erase their ephemeral private keys. This enforces Perfect Forward Secrecy (PFS), ensuring that

compromise of long-term identity keys during the mission cannot reveal the already established Session Master Key. The UAV enters a standby state awaiting the Mission Initialization message in Phase C, which provides explicit key confirmation through MSG4 and MSG5.

4.3 Phase C: Mission Initialization and Key Confirmation

Mission Initialization is executed after successful completion of the Authentication Key Exchange. Its purpose is to establish the first protected post handshake channel state, securely deliver the dynamic mission configuration and provide explicit key confirmation before transitioning to operational traffic. This phase is separate because mission configuration may potentially need to be issued later, when the UAV has already transitioned to BVLOS, and it follows the common secure channel pattern of first establishing an authenticated shared secret and then using derived, purpose-specific keys to protect subsequent channel setup and protected state installation.

To enforce cryptographic separation between the Authentication Key Exchange and the protected mission initialization traffic, dedicated keys are derived from the Session Master Key. The GCS performs this derivation before transmitting the mission initialization message, whereas the UAV derives the same values on demand upon receipt of that message

$SC_{GCS} \leftarrow$ ("Phase C" || MID || PID_{GCS} || PID_{UAV} || TRSCRPT)
$CB_{GCS} \leftarrow H(SC_{GCS})$
$PRKC_{GCS} \leftarrow HKDF_{Extract}(CB_{GCS}, SMK_{GCS})$
$KIN_{GCS} \leftarrow HKDF_{Expand}(PRKC_{GCS}$, "Mission Init Key" || MID, L=128-bit)
$KACK_{GCS} \leftarrow HKDF_{Expand}(PRKC_{GCS}$, "Mission Init ACK key" || MID, L=128-bit)

SYMBOL	DESCRIPTION
SC_X	Shared Context
CB_X (256-bit)	Hashed KDF Context Binder
$PRKC_X$ (256-bit)	Pseudorandom key for this phase
KIN_X (128-bit)	Key used to verify and decrypt mission initialization message
$KACK_X$ (128-bit)	Key used to verify and decrypt UAV acknowledgment

Table 7: Cryptographic Symbols for Specific Purpose Key Generation

The nonce is constructed from public mission and message context. Since the associated counter is monotonically increasing, the same nonce is never repeated under the same key; therefore, it will never be reused. The sender constructs the nonce before encryption, while the receiver reconstructs the same nonce before decryption.

N (D, TP, CTR)← (MID || EID|| D || TP || CNTR)

SYMBOL	DESCRIPTION
N (128-bit)	Nonce
MID (32-bit)	Mission ID
EID (16-bit)	Epoch Identifier
D (8-bit)	Direction (0x00 for GCS to UAV, 0x01 for UAV to GCS)
TP (8-bit)	Msg Type (0x01 for Mission Init MSG_4, 0x02 for Mission Init ACK MSG_5)
CNTR (64-bit)	Monotonically Increasing counter

Table 8: Cryptographic Symbols for Nonce Construction

The GCS constructs and encrypts the **dynamic mission configuration payload** using the initialization key.

$CNTR_{INIT} \leftarrow 1$
$N_{INIT}^{GCS} \leftarrow N$ (0x00, 0x01, $CNTR_{INIT}$)
$PL_{INIT} \leftarrow$ ("Dynamic Mission Parameters" || DMP)
$AD_{INIT}^{GCS} \leftarrow (SC_{GCS} \| 0x01 \| EID)$
$(C_{INIT}, T_{INIT}) \leftarrow$ Ascon-AEAD Encrypt (KIN_{GCS}, N_{INIT}^{GCS}, AD_{INIT}^{GCS}, PL_{INIT})
$MSG_4 \leftarrow$ {"Mission Initialization", MID, EID, $CNTR_{INIT}$, C_{INIT}, T_{INIT}}
Timer $_{LINK} \leftarrow T_{LINK}$ (On timeout retransmit MSG_4)

SYMBOL	DESCRIPTION
AD_X^Y	Associated Data
C_X	Ciphertext
T_X	Ascon Tag

Table 9: Cryptographic Symbols for Ascon-AEAD128

UAV Application of the Rule. The UAV applies the above rule upon receiving MSG_4. Only if all local verification checks succeed does the UAV derive the Phase C keys, reconstruct the corresponding nonce and associated data, and attempt authenticated decryption of the protected payload. Specifically, the UAV first verifies that the received MID matches the current mission, that the received EID is valid for the current epoch, and that the received counter value has not been previously accepted for the same Mission Initialization instance. If any local verification check fails, the UAV silently discards the packet without installing mission state. If all checks succeed, the UAV reconstructs the shared context, derives the same initialization and acknowledgment keys, reconstructs the nonce and associated data, and decrypts the received ciphertext. If authenticated decryption is unsuccessful, the packet is discarded and the UAV returns to passive listening posture. If decryption succeeds, the UAV accepts and installs the dynamic mission parameters.

$SC_{UAV} \leftarrow$ ("Phase C" || MID || PID_{GCS} || PID_{UAV} || TRSCRPT)
$CB_{UAV} \leftarrow H(SC_{UAV})$
$PRKC_{UAV} \leftarrow HKDF_{Extract}(CB_{UAV}, SMK_{UAV})$
$KIN_{UAV} \leftarrow HKDF_{Expand}(PRKC_{UAV}$, "Mission Init Key" || MID, L=128-bit)
$KACK_{UAV} \leftarrow HKDF_{Expand}(PRKC_{UAV}$, "Mission Init ACK key" || MID, L=128-bit)
$N_{INIT}^{UAV} \leftarrow N$ (0x00, 0x01, $CNTR_{INIT}$)
$AD_{INIT}^{UAV} \leftarrow (SC_{UAV} \| 0x01 \| EID)$
$PL_{INIT} \leftarrow$ Ascon-AEAD Decrypt (KIN_{UAV}, N_{INIT}^{UAV}, AD_{INIT}^{UAV}, C_{INIT}, T_{INIT})

After successful installation of the dynamic mission parameters, the UAV constructs the protected acknowledgment message. The transmission of this acknowledgment constitutes explicit key confirmation to the GCS that the UAV derived the correct keying material, successfully decrypted MSG_4 and accepted the transmitted Dynamic Mission Parameters.

$CNTR_{ACK} \leftarrow 1$
$N_{ACK}^{UAV} \leftarrow N$ (0x01, 0x02, $CNTR_{ACK}$)
$PL_{ACK} \leftarrow$ ("DMP Received" || $H(PL_{INIT})$ || ACCEPTED)
$AD_{ACK}^{UAV} \leftarrow (SC_{UAV} \| 0x02 \| EID)$
$(C_{ACK}, T_{ACK}) \leftarrow$ Ascon-AEAD Encrypt ($KACK_{UAV}$, N_{ACK}^{UAV}, AD_{ACK}^{UAV}, PL_{ACK})
$MSG_5 \leftarrow$ {"Mission Initialization Acknowledgement", MID, EID, $CNTR_{ACK}$, C_{ACK}, T_{ACK}}

GCS Application of the Rule. The GCS applies the above rule upon receiving MSG_5. Only if all local verification checks succeed does the GCS reconstruct the corresponding nonce and associated data and attempt authenticated decryption of the protected acknowledgment. Specifically, the GCS first verifies that the received MID matches the current mission, that the received EID is valid for the current epoch and that the received counter value has not been previously accepted for the same Mission Initialization instance. If any local verification

check fails, the GCS discards the packet and continues processing the currently pending initialization state according to the retransmission policy. If all checks succeed, the GCS reconstructs the nonce and associated data and decrypts the received ciphertext. If authenticated decryption is unsuccessful, the acknowledgment is discarded and the Mission Initialization instance remains pending. If decryption succeeds, the GCS verifies that the acknowledgment corresponds to the transmitted initialization payload. Upon successful verification, the GCS marks Mission Initialization as complete and terminates any further retransmissions of MSG_4

$N_{ACK}^{GCS} \leftarrow N\ (0x01, 0x02, CNTR_{ACK})$
$AD_{ACK}^{GCS} \leftarrow (SC_{GCS} \parallel 0x02 \parallel EID)$
$PL_{ACK} \leftarrow$ Ascon-AEAD Decrypt ($KACK_{GCS}$, N_{ACK}^{GCS}, AD_{ACK}^{GCS}, C_{ACK}, T_{ACK})
Verify $H(PL_{INIT}) = H^*(PL_{INIT}) \rightarrow$ From PL_{ACK}

PARAMETER	DESCRIPTION	SECURITY RATIONALE
k (8-bit)	Minimum fragments to reconstruct	AONT-RS reconstruction threshold
n (8)	Total fragments generated	RS redundancy level
tk	Timeout time to wait for k fragments	Head of line blocking
FHSS 1-BIT	FHSS=1 if enabled else FHSS=0	Enabled in contested environments
FHSSPD 8-BIT	Hop Rate	Larger set harder to jam

Table 10: Dynamic Mission Parameters

4.4 Phase D: Key Derivation

Following successful completion of the Mission Initialization phase, both endpoints transform the established mission secret into a structured set of compartmentalised traffic keys. Each Direction–Channel–Link tuple is assigned an independent cryptographic context. This construction limits the blast radius of compromise. Exposure of a key does not imply exposure of associated keys.

For each D, CH, L
SKD (D, CH, L) ← (“Traffic Keys” || MID || EID || D || CH || L)
K [D] [CH] [L] ← $HKDF_{EXPAND}$ (SMK_X, SKD (D, CH, L), l=128-bit)
FHSD ← HKDF_EXPAND (SMKX, “FHSS Seed” || MID || EID, L = 128-bit) (if frequency hopping is enabled)

GCS → UAV	Description	UAV → GCS
K [0x00] [C2] [RF]	Key for C2 on RF link	None
K [0x00] [C2] [Wi-Fi]	Key for C2 on Wi-Fi link	None
K [0x00] [C2] [LTE]	Key for C2 on LTE link	None
K [0x00] [C2] [SAT]	Key for C2 on SAT link	None
K [0x00] [CTRL] [RF]	Key for CTRL on RF link	K [0x01] [CTRL] [RF]
K [0x00] [CTRL] [Wi-Fi]	Key for CTRL on Wi-Fi link	K [0x01] [CTRL] [Wi-Fi]
K [0x00] [CTRL] [LTE]	Key for CTRL on LTE link	K [0x01] [CTRL] [LTE]
K [0x00] [CTRL] [SAT]	Key for CTRL on SAT link	K [0x01] [CTRL] [SAT]
None	Key for TEL on RF link	K [0x01] [TEL] [RF]
None	Key for TEL on Wi-Fi link	K [0x01] [TEL] [Wi-Fi]
None	Key for TEL on LTE link	K [0x01] [TEL] [LTE]
None	Key for TEL on SAT link	K [0x01] [TEL] [SAT]

None	Key for PLD on RF link	K [0x01] [PLD] [RF]
None	Key for PLD on Wi-Fi link	K [0x01] [PLD] [Wi-Fi]
None	Key for PLD on LTE link	K [0x01] [PLD] [LTE]
None	Key for PLD on SAT link	K [0x01] [PLD] [SAT]

Table 11: Key Results

The derivation process produces a total of 20 independent 128-bit traffic protection keys each uniquely associated with a specific Direction–Channel–Link tuple. This corresponds to the Cartesian combination, resulting in an aggregate of 2560 bits of key material. By assigning a distinct cryptographic context to every tuple, the protocol achieves strong compartmentalization, such that compromise of one traffic key does not affect the security of the remaining communication paths. Once all traffic keys have been successfully derived, the Session Master Key (SMK) is securely erased at both endpoints, thereby reducing key retention exposure and limiting the consequences of post-derivation compromise.
After deriving keys, both endpoints initialize an independent monotonically increasing counter. The nonce for each protected transmission is then constructed from the public mission, epoch, tuple context, and the corresponding counter value.
For each Direction (D), Channel (CH), Link (L)
CNTR [D] [CH] [L] ← 1
N (D, CH, L) ← (NP || EID || D || CH || L || CNTR [D] [CH] [L])

Parameter	Purpose
NP	Mission nonce prefix derived from MID and endpoint identities
EID	Epoch identifier
D	Direction identifier
CH	Channel identifier
L	Link identifier
CNTR	Per Direction-Channel-Link monotonic counter
Total	Ascon-AEAD128 nonce length

Table 11a: Phase D Nonce Construction

Here, NP is a 24-bit mission nonce prefix computed as Trunc24(H(MID || PIDGCS || PIDUAV)). It binds the Phase D nonce space to the mission context while keeping the total nonce length at exactly 128 bits, as required by Ascon-AEAD128.
The Associated Data (AD) field binds unencrypted frame metadata to the ciphertext. This prevents an adversary from modifying routing, control, or framing fields without detection.
HDR ← (MID || EID || D || CH || L || CNTR [D] [CH] [L])
AD ← Serialize (HDR)

Each message transmitted by either endpoint is protected using Ascon-AEAD128 authenticated encryption with the corresponding tuple key, constructed nonce, and associated data

(C, T) ← Ascon-AEAD Encrypt (K [D] [CH] [L], N [D] [CH] [L], AD [D] [CH] [L], PL)
PL ← Ascon-AEAD Decrypt (K [D] [CH] [L], N [D] [CH] [L], AD [D] [CH] [L], C, T)

4.5 Phase E: During Flight

Part A: GCS sender UAV receiver

A.1 GCS C2 UAV
The GCS generates a C2 command and assigns a mission-epoch unique command identifier C2MSGID to that logical command before replication. Next, the scheduler iterates over every active communication link L, treating each link independently so that the same command can

be securely transmitted over multiple available paths. For each active link, the protocol first increments the dedicated per-link counter CNTR [0x00] [C2] [L], which ensures freshness and uniqueness for that specific channel-link pair. Using this updated counter, a header is constructed containing the Mission ID (MID), Epoch ID (EID), channel identifier CH, link identifier L, the current counter value, and C2MSGID. This header is then serialized into associated data (AD [L]) that can be used during authenticated encryption. After that, a nonce (N [L]) is deterministically derived from the tuple (Direction, Channel, Link, Counter), guaranteeing a distinct nonce for each transmission on each link. The payload is then encrypted using Ascon-AEAD128 with the per-link key K [0x00] [C2] [L], the derived nonce, and the associated data, producing ciphertext C [L] and authentication tag T [L]. The Epoch Byte Check step ensures that the payload remains bound to the correct epoch context before final transmission. Finally, the complete message MSG [L] = {HDR [L], C[L], T [L]} is assembled and transmitted over link L. In this way, every active link carries its own independently protected version of the same C2 command, with separate counters and nonces but with the same C2MSGID for cross-link deduplication.

On the receiving side, the UAV first receives the packet over link L and parses the corresponding header HDR [L] It then verifies the header tuple (MID, EID, CH, L) to ensure that the message belongs to the correct mission, epoch, channel, and communication link. If this validation fails, the packet is immediately discarded. If the tuple is valid, the UAV extracts the received counter value RCNTR [L] from the header and performs a sliding window check for the tuple (0x00, C2, L). This step provides replay protection by rejecting stale, duplicate, or out-of-window packets before any costly cryptographic processing is performed. For packets that pass this freshness check, the UAV reconstructs the authenticated data AD [L] by serializing the received header and then reconstructs the nonce N [L] using the epoch identifier, direction, channel, link, and received counter value. Using the per-link key K [0x00] [C2] [L], the reconstructed nonce, and the authenticated data, the UAV attempts to decrypt the ciphertext-tag pair (C [L], T [L]). If authentication fails, the packet is discarded because this indicates tampering, corruption, or use of an incorrect cryptographic context. If decryption succeeds, the payload is accepted. Since all replicated copies of the same logical C2 command carry the same C2MSGID, the UAV accepts only the first valid copy for a given C2MSGID and records that identifier in a bounded deduplication cache. Any later valid packet with the same C2MSGID is discarded as a duplicate, while a different C2MSGID is treated as a distinct command even if it arrives with the same per-link counter value on another link.

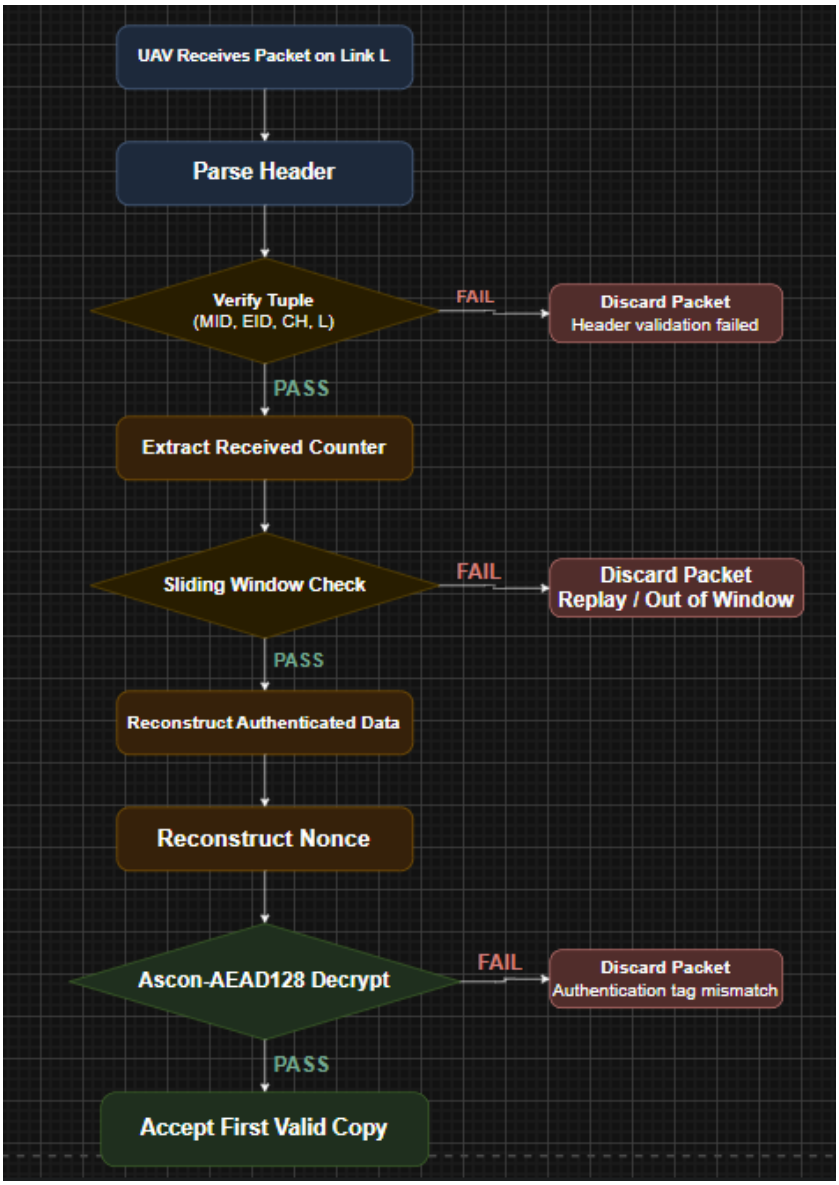


Fig 8. Receiver Process for C2

A.2 GCS CTRL UAV

On the transmission side for the CTRL channel, the procedure begins by generating the control message CTRL from the required control data and assigning it as the payload PL. Before transmission, the payload is processed through a package-transform All-Or-Nothing Transform (AONT). A fresh random package key KAONT is generated using a 128-bit high-quality random source and used to transform the payload. KAONT is not transmitted as a separate field; it is embedded within the AONT package by masking it with a hash of the transformed data, so the receiver can recover KAONT only after enough Reed-Solomon fragments have been collected and the complete AONT package has been reconstructed. This ensures that no useful part of the original payload can be recovered unless the transformed data is reconstructed in full. The AONT output is then encoded using Reed-Solomon coding, producing a fragment set FRGST with parameters (n, k), so that the original transformed payload can later be recovered even if some fragments are lost in transmission. Afterward, the fragments are distributed across the available active links using a round-robin assignment policy, so that successive fragments are spread over different links to improve resilience against link degradation or selective interference. For each fragment f assigned to link L, the protocol increments the corresponding per-link counter CNTR [0x00] [CTRL] [L], constructs

a fragment header HDR [f] containing the Mission ID (MID), Epoch ID (EID), channel identifier, link identifier, fragment identifier FID, fragment-total or fragment-offset field FTO, the current counter value, and the message identifier MSGID, and serializes this header into authenticated data AD [f]. A distinct nonce N [f] is then derived from the tuple (0x00, CTRL, L, CNTR [0x00] [CTRL] [L]), ensuring uniqueness for every encrypted fragment on every link. Using the per-link key K [0x00] [CTRL] [L], the derived nonce, and the authenticated data, the protocol encrypts the assigned Reed-Solomon fragment FRGST [f] with Ascon-AEAD128, producing ciphertext C [f] and authentication tag T [f]. The Epoch Byte Check step binds the fragment to the correct epoch context before transmission. Finally, the complete protected fragment message MSG [f] = {HDR [f], C [f], T [f]} is assembled and transmitted over its assigned link. In this way, the CTRL message is not sent as a single unit, but as multiple independently protected fragments spread across several links, thereby combining confidentiality, integrity, replay resistance, erasure tolerance, and multi-link robustness.

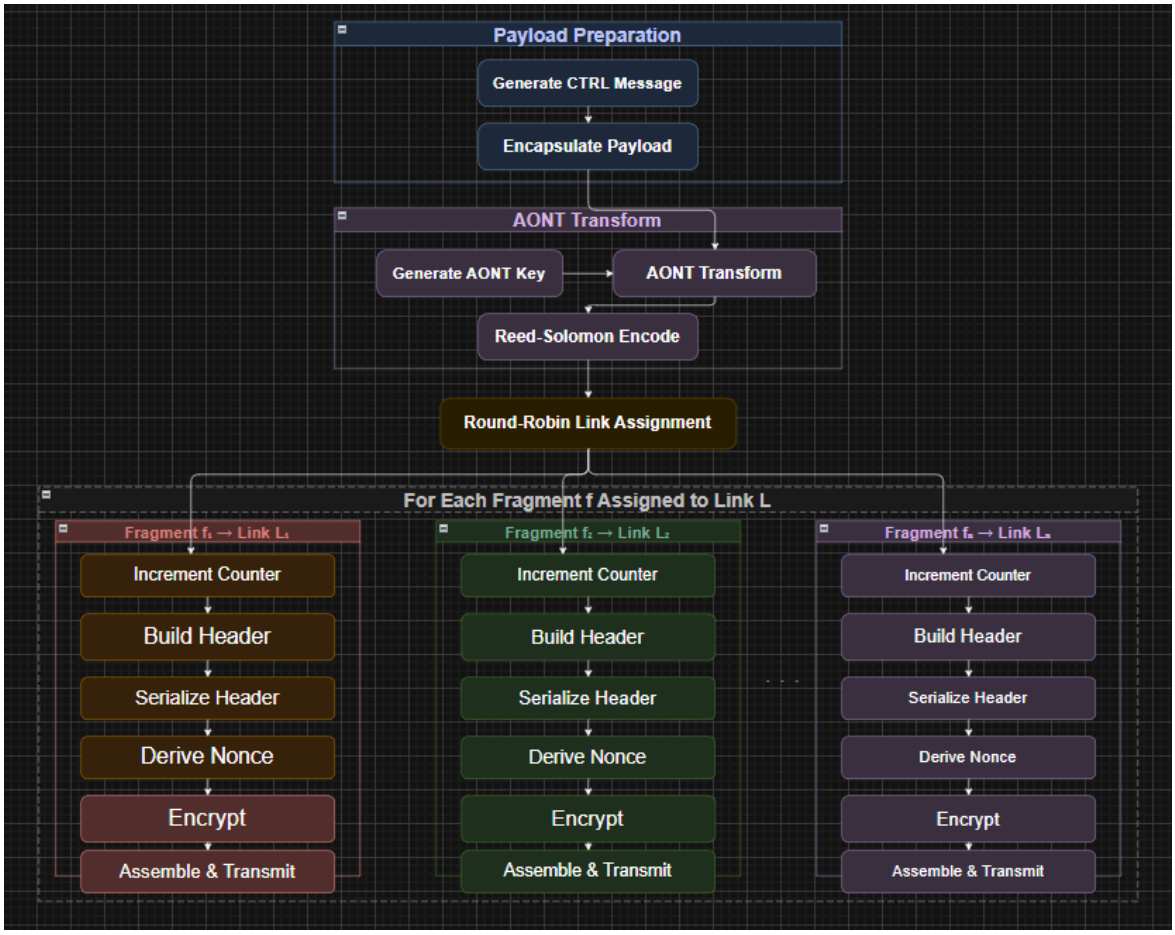


Fig 9. Sender Process for CTRL

On the receiving side for fragmented delivery, the UAV processes each incoming fragment independently as it arrives on a given link. For every received fragment, the UAV first parses the header and verifies the tuple (MID, EID, CH, L) to confirm that the fragment belongs to the correct mission, epoch, channel, and link context. Fragments with invalid header context are immediately silently discarded. If the header is valid, the UAV extracts the received counter and applies a sliding window freshness check, thereby rejecting replayed, duplicated, or out-of-window fragments before decryption. For valid and fresh fragments, the UAV reconstructs the associated data from the serialized header, reconstructs the nonce from the epoch, direction byte, channel, link, received counter, and then attempts to decrypt using the corresponding per-link key. If authentication fails, the fragment is discarded. Otherwise, the recovered plaintext fragment is stored in the reassembly buffer associated with the corresponding message identifier MSGID. After each successful insertion, the UAV checks whether enough fragments have been collected to permit reconstruction. If enough fragments

are available before the await timer expires, Reed-Solomon decoding is applied to recover the AONT protected object, after which the inverse AONT operation reconstructs the original payload, which is then delivered to the UAV processing layer. If sufficient fragments are not received in time, the timer expires, the incomplete reassembly buffer is released, and the message is treated as unsuccessfully reconstructed.

Once the UAV determines the outcome of reassembly, it constructs a protected **ACK/NACK response**. To do so, it increments the appropriate counter, creates an ACK or NACK payload that reflects the reconstruction result for the corresponding MSGID, builds the response header, serializes that header as associated data, derives a fresh nonce from the response context, and encrypts the response under the appropriate per-link response key. After the epoch consistency check, the UAV assembles the final protected ACK/NACK packet and transmits it back to the sender. In this way, the response path provides authenticated feedback on whether fragmented delivery and payload reconstruction were successful, while preserving confidentiality, integrity, replay protection, and epoch consistency.

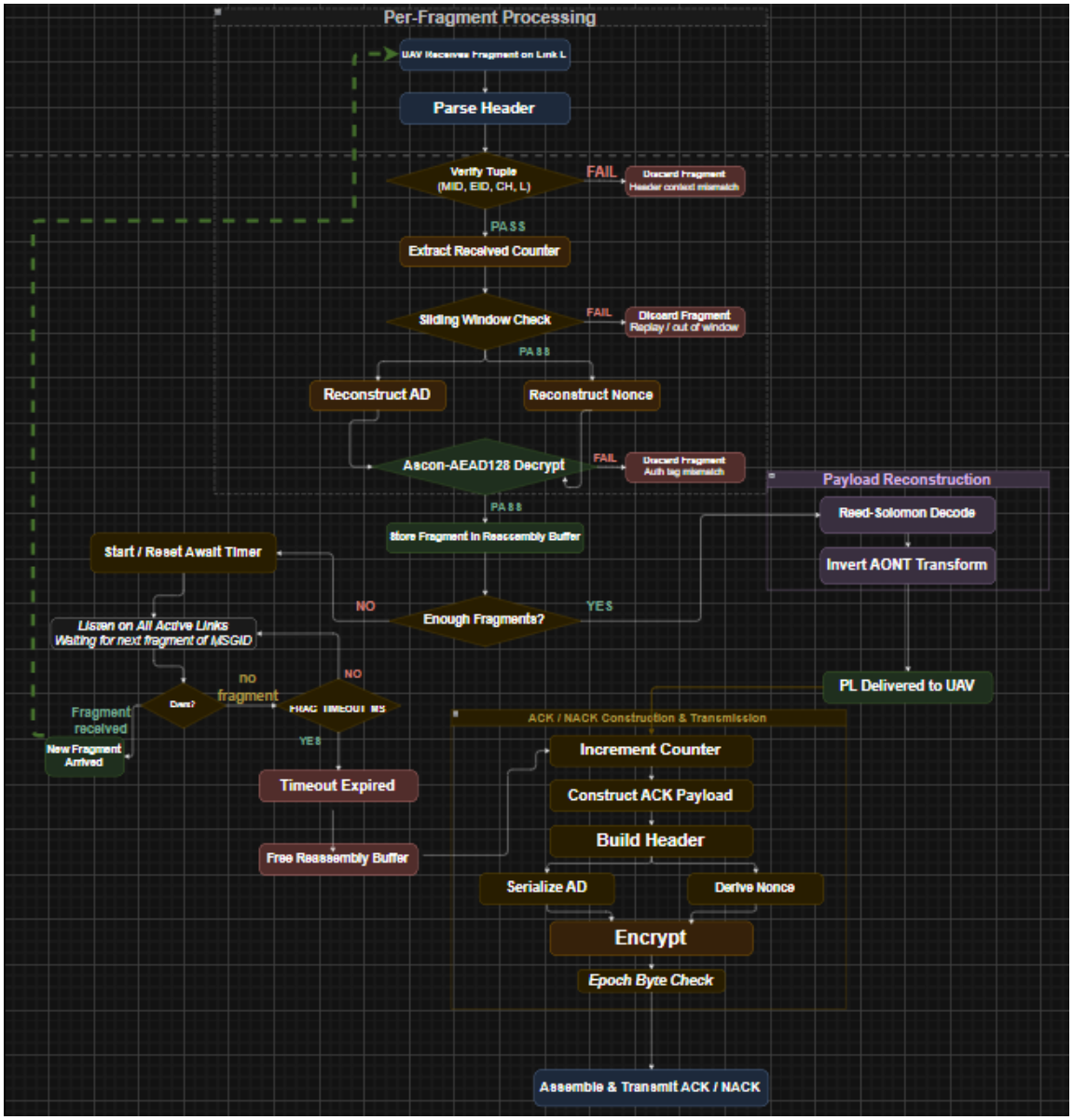


Fig 10. Receiver Process for CTRL

On the feedback reception side, the GCS receives the protected ACK/NACK packet on link L and first parses its header. It then verifies the tuple (MID, EID, CH, L) to confirm that the response belongs to the correct mission, epoch, channel, and link context. If this contextual validation fails, the ACK/NACK is discarded immediately. If the tuple is valid, the GCS

extracts the received counter from the header and performs a sliding window check to reject replayed, duplicated, or out-of-window feedback packets before proceeding to cryptographic processing. For packets that pass this freshness test, the GCS reconstructs the associated data by serializing the received header and reconstructs the nonce from the epoch, direction context, channel, link, and received counter. Using the corresponding per-link key, the GCS then attempts to decrypt the ciphertext and authentication tag. If decryption fails, the ACK/NACK is discarded because this indicates tampering, corruption, or an invalid cryptographic context. If decryption succeeds, the GCS verifies the payload hash carried in the ACK/NACK against the locally stored hash of the original transmitted message, thereby ensuring that the received feedback corresponds to the intended protected message instance. If the hash does not match, the ACK/NACK is rejected. If the hash matches, the GCS inspects the response type if the packet contains an ACK, the message is marked as successfully delivered and the retransmission timer is cancelled; if the packet contains a NACK, the message is treated as unsuccessfully delivered and is scheduled for retransmission.

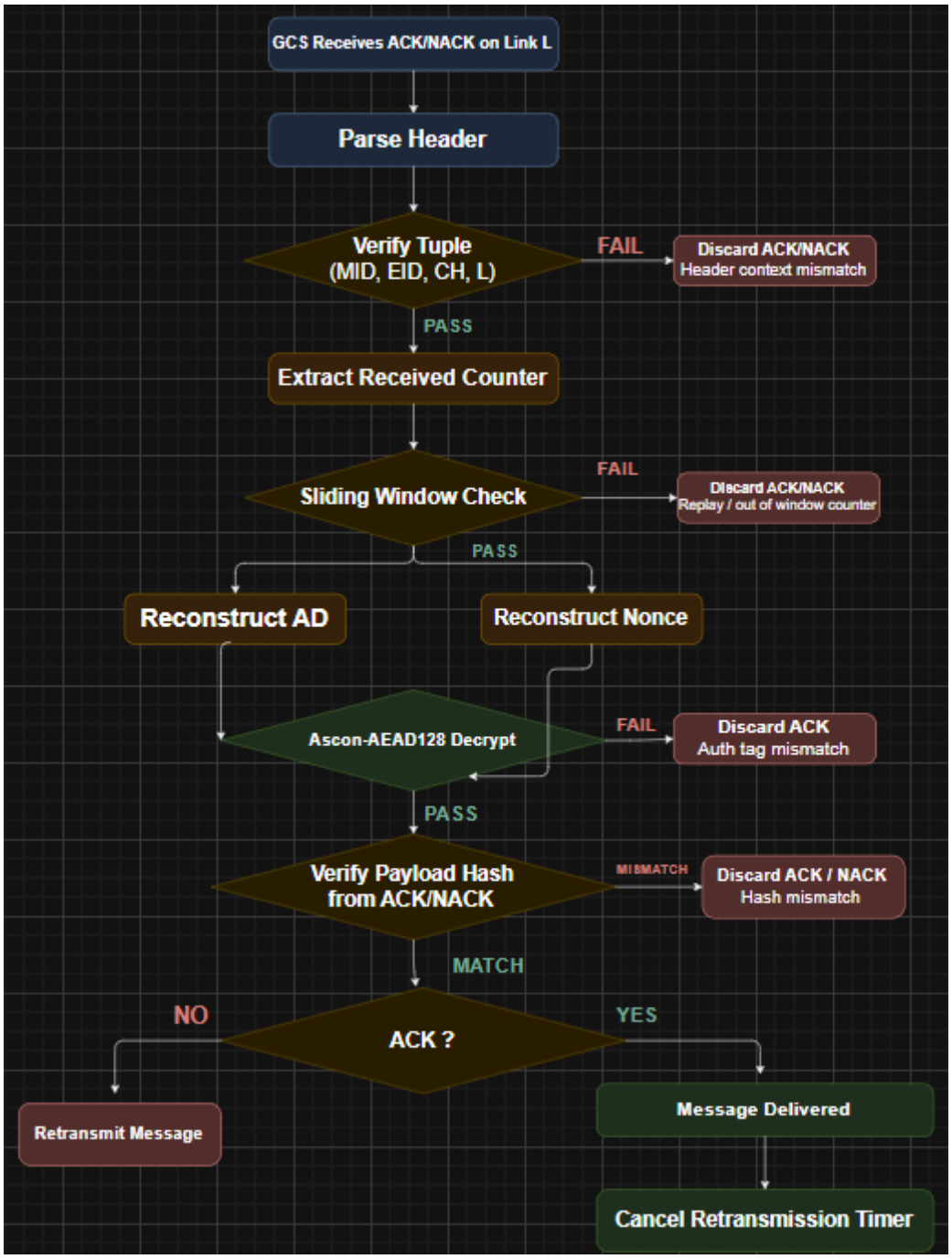


Fig 11. ACK process

Part B: UAV sender GCS receiver

B.1 UAV CTRL GCS

Follows the same operational and security procedure as the GCS CTRL UAV communication process.

B.2 UAV PLD GCS

For large payload delivery, the sender first generates the payload and encapsulates it into the protocol-defined message format. Because the payload may be too large or too important to transmit as a single protected unit, it is first split into multiple chunks, with each chunk treated as an independent cryptographic object. For every chunk c, the protocol generates a dedicated chunk key and derives a corresponding chunk nonce, which are then used to encrypt the chunk individually. After encryption, Reed–Solomon forward error correction is applied to the resulting ciphertext, producing multiple fragments for that chunk so that recovery remains possible even if some fragment transmissions are lost or corrupted. Once this is done for all chunks, the produced fragments are distributed across the available active links using a round-robin link assignment strategy, thereby spreading the transmission load and improving resilience against link degradation, selective jamming, or temporary path failure. For each fragment f of chunk c assigned to link L, the sender increments the corresponding per-link counter, constructs a header containing the relevant context and fragment metadata, serializes the header as authenticated data, derives a fresh per-fragment nonce, and encrypts the fragment for transmission on that link. After encryption, the epoch byte check is applied to each fragment to ensure epoch consistency before final assembly. The protected packet is then assembled and transmitted over the assigned link. In this way, the protocol introduces a two-layer protection structure: first, each chunk is independently encrypted and FEC-protected, and second, each resulting fragment is again protected with per-link authenticated encryption before transmission. This design strengthens confidentiality, integrity, replay resistance, loss tolerance, and multi-link robustness, while also ensuring that large payloads can be delivered efficiently and reconstructed reliably in adversarial communication environments

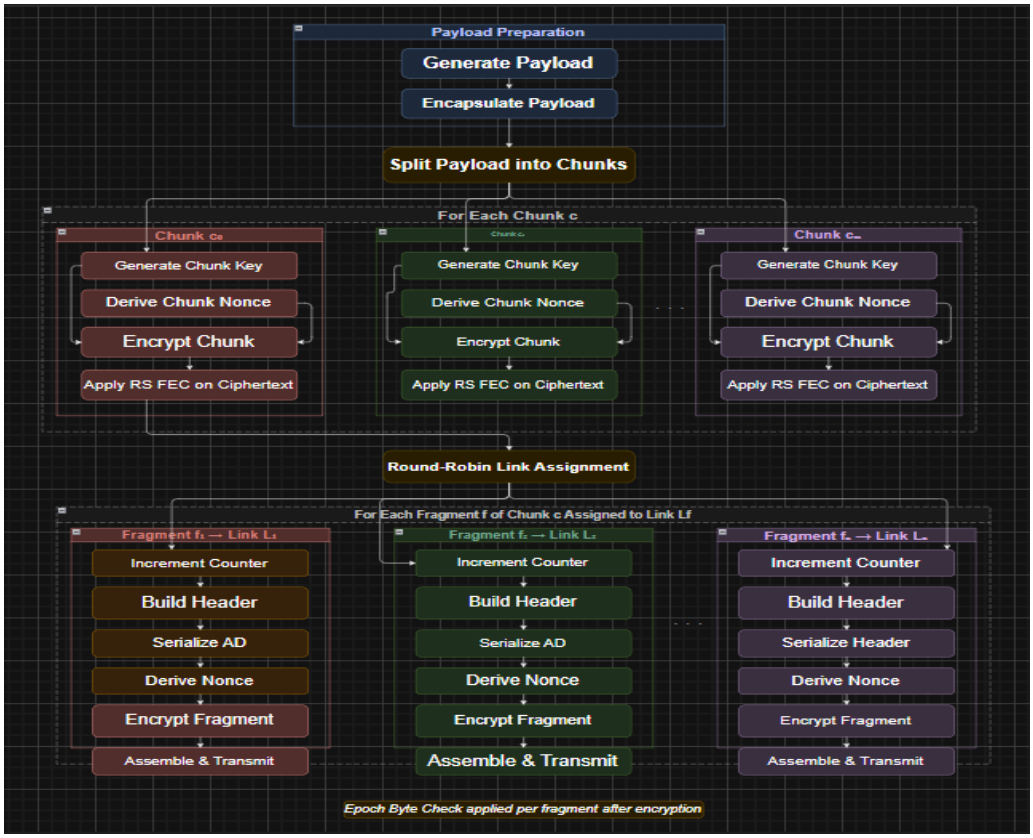


Fig 12. Sender Process for PLD

On the receiving side for chunk-based payload delivery, the GCS processes each incoming fragment independently as it arrives on link L. For every received fragment, the protocol first parses the header and verifies the tuple (MID, EID, PLD, L) to confirm that the fragment belongs to the correct mission, epoch, payload channel, and link context. Any fragment with invalid contextual metadata is discarded immediately. If the tuple is valid, the received counter is extracted and checked using the sliding-window mechanism, which filters out replayed, duplicated, or out-of-window fragments before further processing. For fragments that pass this freshness test, the GCS reconstructs the associated data from the received header, reconstructs the nonce, and performs the outer-layer decryption using the corresponding per-link key. If authentication fails, the fragment is discarded. Otherwise, the recovered FEC fragment is stored in the reassembly buffer for its corresponding chunk. Once enough FEC fragments has been collected for a given chunk, Reed–Solomon decoding is applied to recover the protected chunk ciphertext. The receiver then reconstructs the chunk nonce, retrieves the appropriate chunk key, and performs the inner-layer decryption of the chunk. If this inner decryption fails, the recovered chunk is rejected, otherwise, the plaintext chunk is accepted and stored. This process continues until all chunks of the original payload have been successfully recovered. If all required chunks are available, the complete payload is reassembled and delivered to the GCS. If enough fragments for a chunk are not received before the corresponding await timer expires, the incomplete reassembly state is released, and the missing chunk prevents full payload reconstruction. In this way, the receiver combines per-fragment replay protection and authentication, FEC-based loss recovery, and per-chunk inner decryption to ensure that only authentic, correctly reconstructed, and fully recovered payload data is delivered to the upper layer.

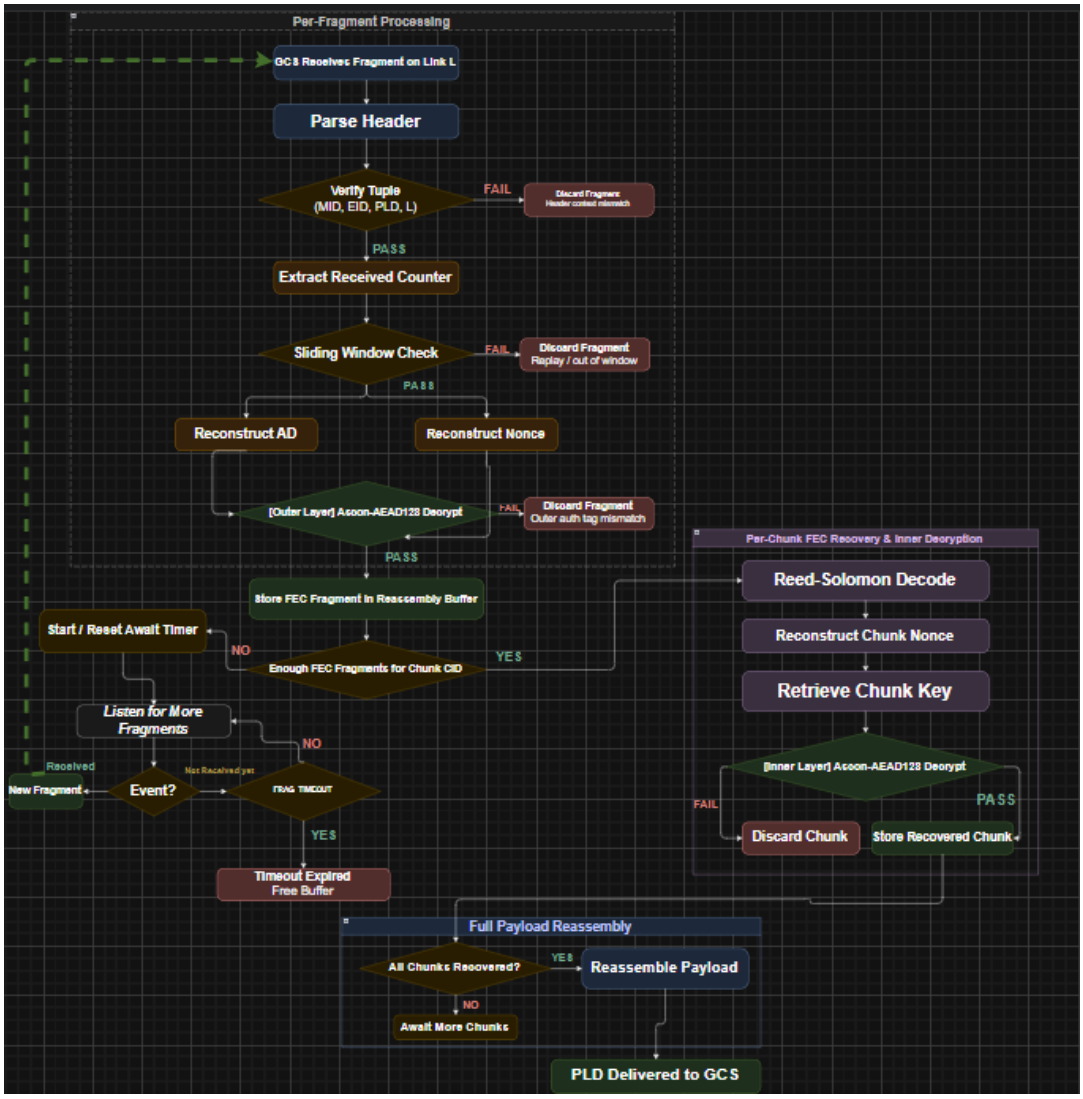


Fig 13. Receiver Process for PLD

B.3 UAV TEL GCS

The UAV first generates the telemetry data and encapsulates it into the protocol-defined payload format. Since telemetry is typically time-sensitive and benefits from high delivery probability rather than reconstruction from fragments, the scheduler replicates the same protected payload across all currently active links. For each active link L, the protocol independently increments the corresponding per-link counter, constructs a header containing the mission, epoch, channel, link, and counter context, and serializes this header as associated data AD. A fresh nonce is then derived for that specific link using the corresponding transmission context, ensuring nonce uniqueness across links and successive telemetry packets. The encapsulated telemetry payload is then encrypted using Ascon under the per-link key, with the derived nonce and serialized header bound into the authenticated encryption process. After encryption, the epoch byte check is applied per link to confirm that the telemetry payload remains correctly bound to the current epoch context before transmission. Finally, the complete protected telemetry message, consisting of the header, ciphertext, and authentication tag, is assembled and transmitted on each active link. In this way, the same telemetry information is sent as multiple independently protected copies, each with its own counter, nonce, and cryptographic protection, thereby improving resilience against packet loss, link degradation, or selective interference, while still preserving confidentiality, integrity, authenticity, replay protection, and epoch consistency in the multi-link adversarial environment

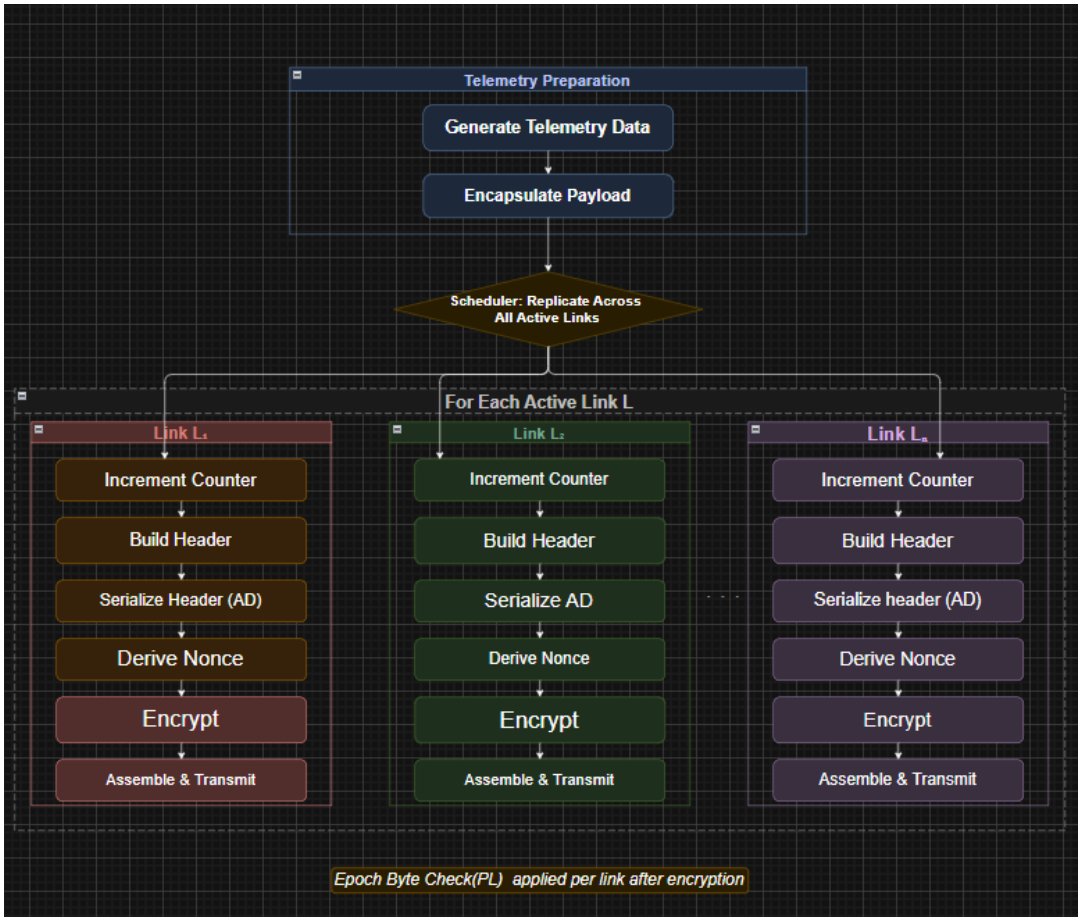


Fig 14. Sender Process for TEL

The GCS processes each incoming telemetry packet independently as it arrives on link L. For every received packet, the protocol first parses the header and verifies the tuple (MID, EID, TEL, L) to ensure that the message belongs to the correct mission, epoch, telemetry channel, and communication link. If this contextual validation fails, the packet is immediately discarded. If the tuple is valid, the GCS extracts the received counter and applies the sliding-window check to reject replayed, duplicated, or out-of-window telemetry packets before any costly cryptographic processing is performed. For packets that pass this freshness check, the GCS reconstructs the associated data from the serialized header, reconstructs the nonce from the epoch, direction context, telemetry channel, link identifier, and received counter, and then performs Ascon decryption using the corresponding per-link key. If authentication fails, the packet is discarded because this indicates tampering, corruption, or an invalid cryptographic context. If decryption succeeds, the telemetry payload is accepted as authentic and fresh, and because the same telemetry message may have been replicated across multiple active links, the GCS accepts only the first valid copy while discarding later duplicates. After acceptance, the recovered telemetry sample is forwarded to the LSTM-based anomaly detection pipeline, where it is appended to a sliding analysis window. The LSTM autoencoder then reconstructs this telemetry window, the reconstruction error is computed, and this error is transformed into an anomaly score. Finally, the anomaly score is compared against a predefined threshold to determine whether the received telemetry behaviour is consistent with normal mission operation or indicative of a possible anomaly. In this way, the reception pipeline not only ensures confidentiality, integrity, authenticity, replay protection, and duplicate suppression for telemetry delivery, but also immediately feeds trusted telemetry into a higher-layer anomaly-detection mechanism for mission monitoring and security analysis

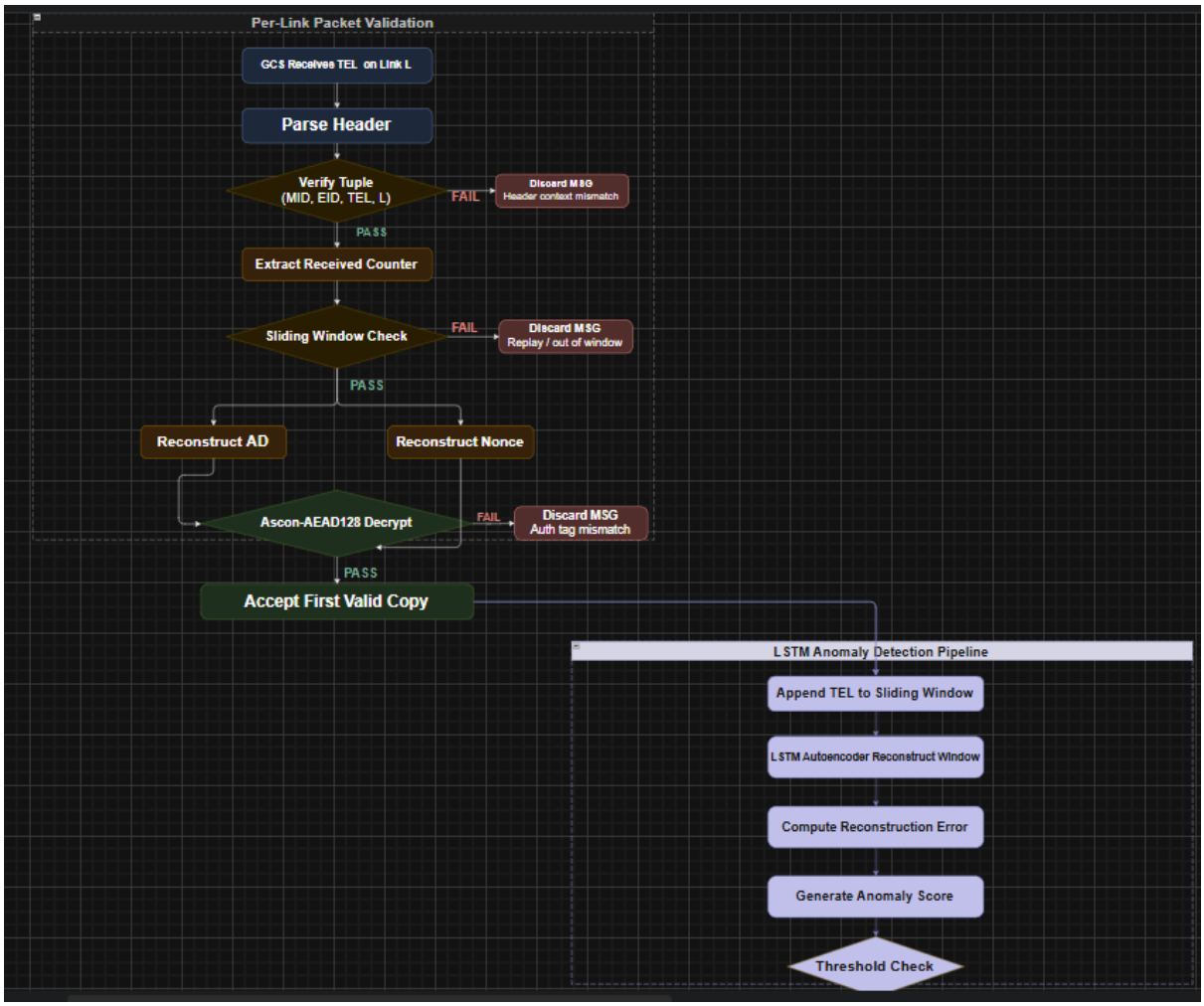


Fig 15. Receiver Process for TEL

Epoch Rotation Procedure

The epoch rotation procedure provides controlled forward key evolution for all per-direction, per-channel, and per-link traffic keys. When rotation is triggered, the protocol first computes a new epoch identifier (EIDNEW = EIDOLD + 1). It then iterates over every tuple (D, CH, L) and constructs an epoch-specific derivation context by concatenating the domain-separation label "Epoch Rotation" with the mission identifier (MID), the new epoch identifier (EIDNEW), the direction D, the channel CH, and the link L. Using this context, a new key KNEW [D] [CH] [L] is derived from the current key K [D] [CH] [L] through HKDF, thereby ensuring that each updated key is cryptographically bound to the new epoch and uniquely separated across direction, channel, and link. After derivation, the newly generated keys are installed by first copying the current keys into KOLD [D] [CH] [L] and then replacing the active keys. At the same time, the corresponding per-tuple counters are reset to 1 so that the new epoch begins with a fresh nonce space. Once all tuples have been updated, the protocol resets the epoch-wide accounting variables by setting the encrypted-byte counter to zero, recording the current time and updating the active epoch identifier to. To prevent loss of in-flight packets during the transition, the previous keys are retained for a grace interval of duration, during which packets protected under K OLD may still be accepted. When the grace timer expires, all K OLD entries are deterministically zeroized from memory. In this way, epoch rotation ensures bounded key lifetime, clean epoch separation, controlled overlap for transition safety, and forward-only symmetric key progression across mission execution. This procedure provides backward secrecy for previous epoch keys after zeroization, but it does not provide forward secrecy for future traffic keys if the current traffic key is compromised, because future epoch keys are derived from the current key. True forward secrecy is provided by the Phase B AKE with ephemeral keys; recovery from a current traffic-key compromise requires a fresh AKE and a new Session Master Key before deriving future traffic keys.

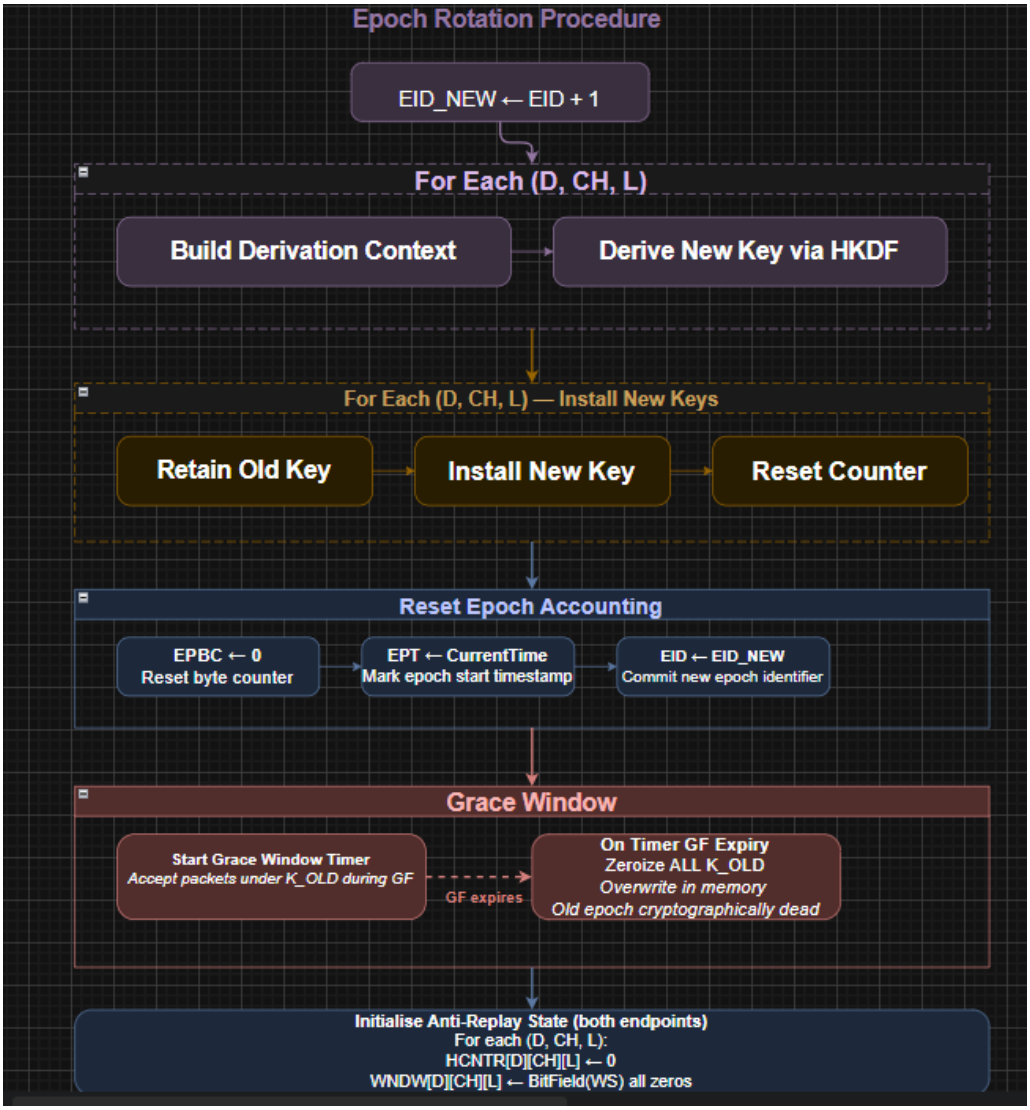


Fig 16. Epoch Rotation Procedure

Sliding Window Verification Procedure

The sliding window verification procedure is performed after header validation and is responsible for determining whether the received packet is fresh, reordered but still acceptable, duplicated, or too old to trust. The procedure operates on the received counter value (RCNTR) and the receiver's maintained anti-replay state for the corresponding tuple (D, CH, L), namely the highest accepted counter HCNTR[D] [CH] [L] and the bitmap window WNDW [D] [CH] [L]. If received counter is bigger than the highest counter, the packet is considered newer than any previously accepted packet on that stream. In this case, the sliding window is advanced by (RCNTR - HCNTR), the received counter is marked as seen in the bitmap, HCNTR is updated to RCNTR, and the packet is accepted. If this condition does not hold, the receiver checks whether the counter still falls within the acceptable replay window. If the counter lies within this range, the packet may simply be delayed or reordered. The receiver then checks whether that counter value is already marked in the bitmap. If it has already been recorded, the packet is discarded as a replay or duplicate. Otherwise, the counter is marked in the bitmap and the packet is accepted. Finally, if the counter does not satisfy either of the previous cases, it is considered too old and therefore outside the valid replay window, so the packet is discarded immediately. In this way, the sliding window mechanism provides efficient replay protection while still tolerating bounded packet reordering, allowing fresh out-of-order packets to be accepted without requiring decryption of obviously stale or duplicated traffic

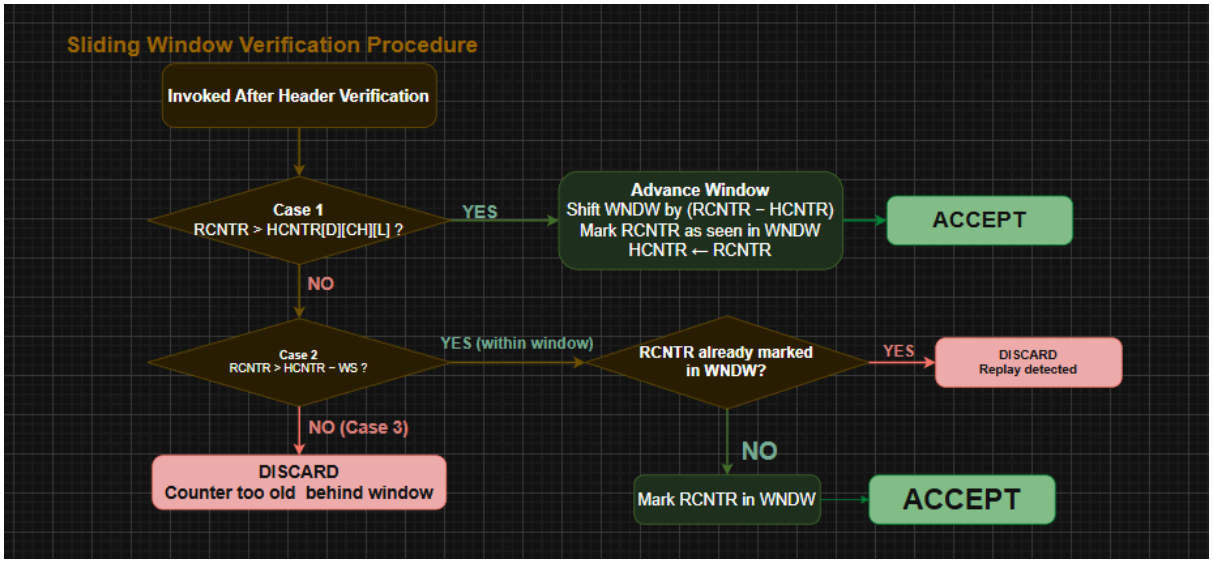


Fig 17. Sliding Window Verification Procedure

Epoch Byte/ Time Check Procedure

The epoch byte and time check procedure is invoked immediately after each encryption operation to enforce bounded key usage within the current epoch. Its input is the payload that has just been encrypted under the active epoch key. First, the protocol increments the epoch byte counter by the size of the encrypted payload. This byte counter tracks the total amount of data protected under the current epoch key set. After updating the counter, the protocol evaluates whether either of the two epoch-lifetime limits has been exceeded: the byte bound RB, which caps the total encrypted volume under the current epoch, or the time bound RT, which limits how long the current epoch may remain active. If neither condition holds, transmission continues normally and no further action is required. However, if either the byte threshold or the time threshold is reached, the protocol immediately triggers epoch rotation. In this way, the procedure ensures that no epoch key remains in use for excessive data volume or excessive duration, thereby maintaining compliance with AEAD usage bounds, reducing long-term cryptanalytic exposure, and enforcing periodic forward progression of the epoch-based key schedule

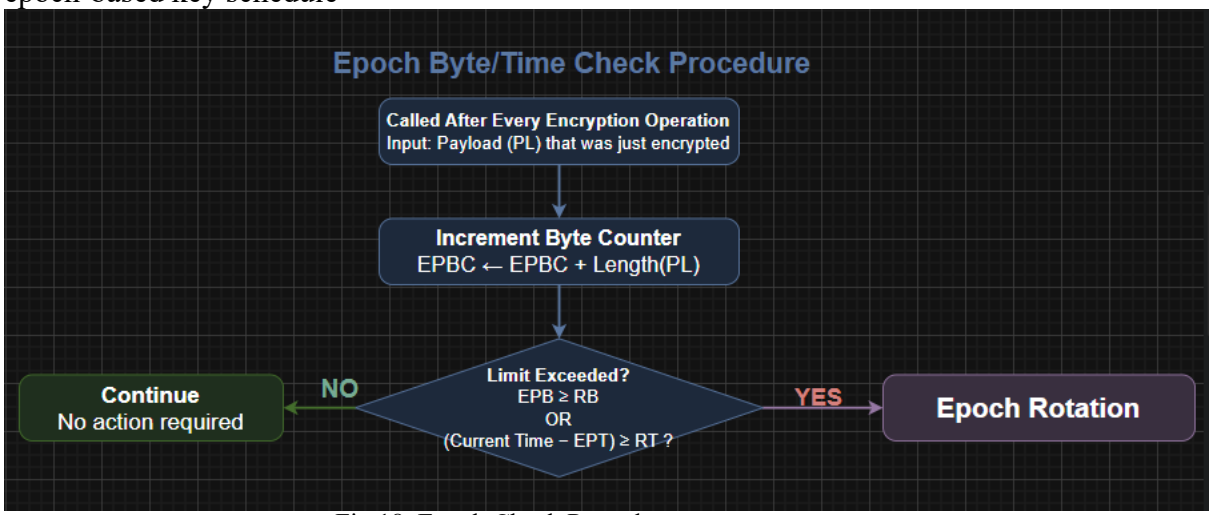


Fig 18. Epoch Check Procedure

The scheduler assigns a configurable transmission priority to each message type according to the operational objectives of the current mission. In this way, the scheduler supports both flexibility and mission-awareness, allowing communication resources to be aligned with the UAV's immediate objectives and risk environment.

The channel semantics are defined as follows. C2 carries flight-critical command-and-control instructions from the GCS to the UAV, such as route changes, altitude changes, return-to-home commands, and immediate mission commands. CTRL carries protocol-control and management messages, such as signed database replica updates, rekeying instructions, configuration updates, ACK/NACK feedback, and other control-plane messages. Therefore, C2 is optimized for low-latency replicated delivery with C2MSGID-based deduplication, while CTRL is optimized for integrity-preserving fragmented delivery using AONT-RS.

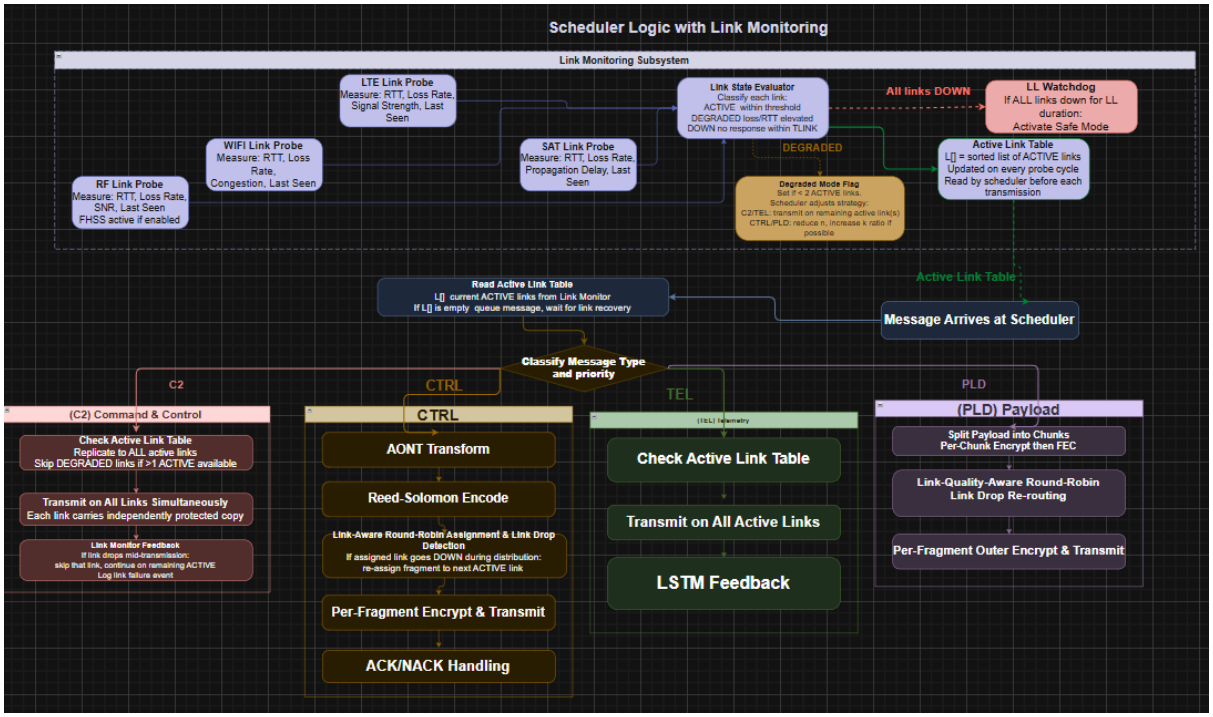


Fig 19. Schedulers Logic

Data Type	Description	Goal	Strategy
C2	Flight-critical C2 instructions	Low latency and reliability	Replication + C2MSGID
CTRL	Protocol-control messages	Integrity and reconstruction	AONT-RS
TEL	Telemetry	Efficiency	Replication + LSTM
PLD	Payload	Bandwidth	RS-FEC

Table 12: Schedulers Logic

4.6 Phase F: Mission End

At mission termination, the GCS issues an authenticated termination command that instructs the UAV to cease mission operations, initiate Return-to-Home, and enter a secure teardown state. In this state, mission-scoped volatile protocol state and session-derived cryptographic material are zeroized from RAM, while mission data retained on persistent storage is sanitized through cryptographic erase by destroying the corresponding mission-specific storage key. This design is consistent with prior work on UAV remote erasure and with established media-sanitization guidance for key-destruction-based sanitization. Long-term TPM-protected identity credentials are retained during routine mission completion and are removed only in a secure environment.

Chapter 5

Conclusion

5.1 Limitations

The proposed contribution has several limitations that should be acknowledged. First, it has not yet been implemented and tested on a real UAV platform under realistic flight conditions. Although the goal was to address secure UAV communication, its practical performance has not yet been validated. In addition, it does not fully evaluate the scalability of the protocol in large, decentralized UAV swarms. Moreover, it still introduces additional computation and communication overhead because it relies on cryptographic operations, authenticated encryption, key derivation and multi-link message protection.
Finally, a future concern is the impact of quantum computing on the cryptographic mechanisms used by the protocol. Since the design depends on modern public-key cryptography for authentication and key exchange, future versions should consider post-quantum cryptographic alternatives or hybrid cryptographic.

5.2 Future Work

Future work should focus on implementing the complete protocol and evaluating it under realistic UAV operating conditions. A practical implementation on real UAV hardware would allow the protocol to be tested during actual flight scenarios, including normal communication, degraded links, packet loss, delay, and adversarial conditions. The protocol should also be evaluated in larger UAV network settings, especially decentralized swarms, to understand how the trust-management, key-derivation, and multi-link communication mechanisms scale as the number of UAVs increases. Finally, future work should examine possible optimizations to reduce computational cost, communication overhead, memory usage, and energy consumption, especially for resource-constrained UAV platforms.

REFERENCES

[1] L. R. Newcome, *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2004.
[2] Imperial War Museums, “A Brief History of Drones.”
[3] National Museum of the United States Air Force, “Kettering Aerial Torpedo ‘Bug’.”
[4] J. D. Blom, *Unmanned Aerial Systems: A Historical Perspective*. Fort Leavenworth, KS, USA: Combat Studies Institute Press, U.S. Army Combined Arms Center.
[5] C. Cuerno-Rejado, L. Garcia-Hernandez, A. Sanchez-Carmona, A. Carrio, J. L. Sanchez-Lopez, and P. Campoy, “Evolution of the unmanned aerial vehicles until present,” 2016.
[6] Encyclopaedia Britannica, “Military aircraft: Unmanned aerial vehicles.”
[7] Encyclopaedia Britannica, “RQ-4 Global Hawk.”
[8] T. P. Ehrhard, *U.S. Unmanned Aerial Vehicles in Combat, 1991–2003*. U.S. Air Force History and Museums Program.
[9] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, “A compilation of UAV applications for precision agriculture,” *Computer Networks*, vol. 172, 2020.
[10] European Commission, “UAV systems for civilian applications,” 2015.

[11] Food and Agriculture Organization of the United Nations, “Exploring agricultural drones: The future of farming, precision agriculture, mapping and spraying,” 2017.
[12] Food and Agriculture Organization of the United Nations and International Telecommunication Union, *E-agriculture in Action: Drones for Agriculture*.
[13] B. Mendu, “State-of-the-Art Review on the Application of Unmanned Aerial Vehicles for Power Line Inspection,” *Drones*, vol. 9, no. 4, 2025.
[14] R. Bielawski, “Unmanned Aerial Vehicles in the protection of the elements of a country’s critical infrastructure,” *Security and Defence Quarterly*, 2018.
[15] F. Outay, H. A. Mengash, and M. Adnan, “Applications of unmanned aerial vehicle in road safety, traffic and highway infrastructure management: Recent advances and challenges,” *Transportation Research Part A: Policy and Practice*, 2020.
[16] M. C. Hunter, “First Responder UAS Use in Post-Disaster Environments,” NASA Technical Reports Server, 2025.
[17] K. Ellis, “NASA Research to Expand UAS Operations for Disaster Response,” ICAS, 2024.
[18] NASA, “What is Unmanned Aircraft Systems Traffic Management?” 2021.
[19] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, “Towards the unmanned aerial vehicles: A comprehensive review,” *Drones*, vol. 6, no. 6, 2022.
[20] M. Ghamari, P. Rangel, M. Mehrubeoglu, G. S. Tewolde, and R. S. Sherratt, “Unmanned aerial vehicle communications for civil applications: A review,” *IEEE Access*, vol. 10, pp. 102492–102531, 2022.
[21] International Civil Aviation Organization, *Unmanned Aircraft Systems Traffic Management (UTM): A Common Framework with Core Principles for Global Harmonization*, 4th ed. Montreal, QC, Canada: ICAO, 2023.
[22] P. Mykytyn, M. F. Khan, and A. Ferworn, “A Survey on Sensor- and Communication-Based Issues of Unmanned Aerial Vehicles,” *Journal of Network and Computer Applications*, vol. 219, 2023.
[23] M. P. Stewart and S. T. Martin, “Unmanned aerial vehicles: Fundamentals, components, mechanics, and regulations,” in *Unmanned Aerial Vehicles*, Nova Science Publishers, 2020.
[24] X. Yu and Y. Zhang, “Architecture, classification, and applications of contemporary unmanned aerial vehicles,” *IEEE Consumer Electronics Magazine*, vol. 11, no. 1, pp. 9–20, 2022.
[25] K. A. A. N. B. Lakew, U. Sa’ad, N. Dao, W. Na, and S. Cho, “Routing in flying ad hoc networks: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1071–1120, 2020.
[26] X. Cao, P. Yang, M. Alzenad, X. Xi, D. Wu, and H. Yanikomeroglu, “Airborne communication networks: A survey,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1907–1926, 2018.
[27] X. Zhang, J. Chen, Y. Li, and L. Wang, “Research on UAV swarm network modeling and resilience assessment,” *Drones*, vol. 7, no. 12, 2023.
[28] İ. Bekmezci, O. K. Sahingoz, and Ş. Temel, “Flying Ad-Hoc Networks (FANETs): A survey,” *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.
[29] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, M. B. Yagoubi, and M. L. B. Yagoubi, “Routing in Flying Ad Hoc Networks: Survey, constraints, and future challenge perspectives,” *IEEE Access*, vol. 7, pp. 81057–81105, 2019.
[30] “Iran–U.S. RQ-170 incident,” public reporting summary.

[31] S. Peterson, “Downed US drone: How Iran caught the ‘beast’,” *The Christian Science Monitor*, Dec. 2011.
[32] Center for Strategic and International Studies, “The Russia-Ukraine Drone War: Innovation on the Frontlines and Beyond,” 2025.
[33] D. Hambling, “How Ukraine’s Killer Drones Are Beating Russian Jamming,” *IEEE Spectrum*, 2025.
[34] Business Insider, “Russia is leveling up its Shahed drones with new Chinese antennae to defeat Kyiv’s jammers, Ukrainians say,” 2025.
[35] L. Buchanan, “The invisible Russia-Ukraine battlefield,” *Wired*, 2024.
[36] A. Shafique, A. Mehmood, and M. Elhadef, “Survey of security protocols and vulnerabilities in unmanned aerial vehicles,” *IEEE Access*, vol. 9, pp. 46927–46948, 2021.
[37] K. Y. Tsao, T. Girdler, and V. G. Vassilakis, “A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks,” *Ad Hoc Networks*, vol. 133, 2022.
[38] R. Aissaoui, J. L. Beuchat, and J. M. Dricot, “A survey on cryptographic methods to secure communications for unmanned aerial vehicles,” 2023.
[39] A. Patel and A. K. Cherukuri, “Analysis of light-weight cryptography algorithms for UAV-networks,” arXiv preprint arXiv:2504.04063, 2025.
[40] S. Sarkar, A. Roy, and S. Misra, “Secure Communication in Drone Networks: A Survey,” *Drones*, vol. 9, no. 8, 2025.
[41] M. Sönmez Turan, K. McKay, D. Chang, J. Kelsey, and J. M. Schanck, *Ascon-Based Lightweight Cryptography Standards for Constrained Devices*, NIST Special Publication 800-232, National Institute of Standards and Technology, 2025.
[42] Z. He, Y. Zhang, and X. Wang, “Security-enhanced lightweight authentication key-agreement protocol for unmanned aerial vehicle communication,” *Applied Sciences*, vol. 15, no. 9, 2025.
[43] T. Jargalsaikhan, K. Lee, Y.-K. Jun, and S. Lee, “Architectural Process for Flight Control Software of Unmanned Aerial Vehicle with Module-Level Portability,” *Aerospace*, vol. 9, no. 2, Art. no. 62, 2022

Appendix

This appendix presents the formal verification model of Authentication Key Exchange (AKE). The model was written in the Tamarin and is used to analyse the symbolic security properties under a Dolev–Yao adversarial model. The purpose is to verify that the Ground Control Station (GCS) and the UAV can establish a shared **S**ession Master Key (SMK) while satisfying the main security goals.

ATTACK	DESCRIPTION	RELATED LEMMA	RESULT

Session Key disclosure / Eavesdropping	The Session Master Key (SMK) is not learned by the attacker	smk_secrecy	Verified
PFS	Master session key remained protected even if long-term keys are revealed	perfect_forward_secrecy	Verified
UAV impersonation	If GCS completes, a genuine UAV session response must have occurred	mutual_auth_gcs_to_uav	Verified
GCS impersonation	If UAV completes, a genuine GCS session response must have occurred	mutual_auth_uav_to_gcs	Verified
Tampering on MSG1 / MITM on first unsigned message	Any successful competition at UAV is bound to the original GCS ephemeral key sent in MSG_1	msg1_tampering_detection	Verified
Reflection / self-session attack	A party cannot successfully complete a session with itself	no_reflection	Verified
Key mismatch / session disagreement	Both parties derive the same SMK for the same session parameters	key_agreement	Verified
Replay Attack	Successful completed sessions cannot be replayed as distinct valid sessions	replay_protection	Verified
Executability	Execution trace that completes successfully	executability	Verified

```
/* Phase B: Authentication Key Exchange (AKE)
 * Tamarin Prover Formal Verification Model
 */

theory UAV_AKE_Protocol
begin

builtins: signing, diffie-hellman, hashing

functions: hkdf/4     /* hkdf(SS, salt, info_label, mission_id) */
         , pid/3      /* pid(pk, mission_id, serial) */
         , tlv/1      /* TLV encoding wrapper */

/* Register GCS long-term identity
 * Models: key generation, PID computation, database enrollment */

rule Register_GCS:
   let
      pk_gcs   = pk(~sk_gcs)
      pid_gcs  = pid(pk_gcs, ~mission_id, ~serial_gcs)
   in
   [ Fr(~sk_gcs)
   , Fr(~mission_id)
   , Fr(~serial_gcs)
   ]
  --[ OnlyOnce(<'RegisterGCS', pid_gcs, ~mission_id>)
```

```
    , Registered('GCS', pid_gcs, ~mission_id)
    ]->
    [ !GCS_LTK(~sk_gcs, pk_gcs, pid_gcs, ~mission_id, ~serial_gcs)
    , !MissionDB(~mission_id, pid_gcs, pk_gcs, 'GCS')
    , Out(pid_gcs)    /* PID (pseudonymous) */
    , Out(pk_gcs)     /* Public key */
    ]

/*
 * Register UAV long-term identity
 * Models: key generation, PID computation, database enrollment
 */
rule Register_UAV:
    let
        pk_uav   = pk(~sk_uav)
        pid_uav  = pid(pk_uav, mission_id, ~serial_uav)
    in
    [ Fr(~sk_uav)
    , Fr(~serial_uav)
    , !MissionDB(mission_id, pid_gcs, pk_gcs, 'GCS')
    ]
  --[ OnlyOnce(<'RegisterUAV', pid_uav, mission_id>)
    , Registered('UAV', pid_uav, mission_id)
    ]->
    [ !UAV_LTK(~sk_uav, pk_uav, pid_uav, mission_id, ~serial_uav)
    , !MissionDB(mission_id, pid_uav, pk_uav, 'UAV')
    /* UAV has read-only replica containing GCS identity */
    , !UAV_Replica(pid_uav, mission_id, pid_gcs, pk_gcs)
    /* GCS database has UAV identity */
    , !GCS_TrustDB(pid_gcs, mission_id, pid_uav, pk_uav)
    , Out(pid_uav)
    , Out(pk_uav)
    ]

/* PHASE B: Authentication Key Exchange */

/*
 * STEP 1: GCS initiates handshake
 * Generates ephemeral X25519 keypair, nonce, timestamp
 * MSG1 is UNSIGNED
 */
rule GCS_Send_MSG1:
    let
        epk_gcs = 'g'^~esk_gcs    /* X25519 ephemeral public key */
        msg1    = <'HandshakeInit', pid_gcs, pid_uav, ~n_gcs, ~t_gcs, epk_gcs>
    in
    [ !GCS_LTK(sk_gcs, pk_gcs, pid_gcs, mission_id, serial_gcs)
```

```
    , !GCS_TrustDB(pid_gcs, mission_id, pid_uav, pk_uav)
    , Fr(~esk_gcs)
    , Fr(~n_gcs)
    , Fr(~t_gcs)
    ]
  --[ GCS_Sent_MSG1(pid_gcs, pid_uav, mission_id, ~n_gcs, epk_gcs)
    , RunningGCS(pid_gcs, pid_uav, mission_id, ~n_gcs, epk_gcs)
    ]->
    [ Out(msg1)
    , GCS_State_1(sk_gcs, pk_gcs, pid_gcs, pid_uav, pk_uav,
                  mission_id, ~esk_gcs, epk_gcs, ~n_gcs, ~t_gcs)
    ]

/*
 * STEP 2: UAV receives MSG1, applies Verification Rule, responds with
MSG2
 * - Verifies PID_GCS in local replica
 * - Verifies PID_UAV matches own identity
 * - Checks nonce freshness
 * - Only THEN generates ephemeral key (resource conservation)
 * - Signs payload binding BOTH ephemeral keys, nonces, timestamps, and
protocol label
 */
rule UAV_Recv_MSG1_Send_MSG2:
    let
        msg1        = <'HandshakeInit', pid_gcs, pid_uav, n_gcs, t_gcs, epk_gcs>
        epk_uav     = 'g'^~esk_uav
        payload_uav = <'UAV_confirm', pid_uav, pid_gcs, mission_id,
                       epk_uav, epk_gcs, ~n_uav, n_gcs, ~t_uav, t_gcs>
        sign_uav    = sign(payload_uav, sk_uav)
        msg2        = <'HandshakeResp', pid_uav, pid_gcs, ~n_uav, ~t_uav,
epk_uav, sign_uav>
    in
    [ In(msg1)
    , !UAV_LTK(sk_uav, pk_uav, pid_uav, mission_id, serial_uav)
    , !UAV_Replica(pid_uav, mission_id, pid_gcs, pk_gcs)    /* Verify PID_GCS
in replica */
    , Fr(~esk_uav)
    , Fr(~n_uav)
    , Fr(~t_uav)
    ]
  --[ UAV_Sent_MSG2(pid_uav, pid_gcs, mission_id, ~n_uav, n_gcs, epk_uav,
epk_gcs)
    , UAV_Verified_PID(pid_gcs)     /* PID lookup */
    ]->
    [ Out(msg2)
    , UAV_State_1(sk_uav, pk_uav, pid_uav, pid_gcs, pk_gcs,
```

```
            mission_id, ~esk_uav, epk_uav, epk_gcs,
            ~n_uav, n_gcs, ~t_uav, t_gcs)
  ]

/*
 * STEP 3: GCS receives MSG2, applies Verification Rule, verifies signature,
sends MSG3
 * - Verifies PID_UAV in trusted database
 * - Checks nonce freshness
 * - Verifies UAV signature over full payload
 * - Signs own payload binding all exchanged parameters
 */
rule GCS_Recv_MSG2_Send_MSG3:
  let
    msg2       = <'HandshakeResp', pid_uav, pid_gcs, n_uav, t_uav, epk_uav,
sign_uav>
    /* Reconstruct expected UAV payload for signature verification */
    payload_uav = <'UAV_confirm', pid_uav, pid_gcs, mission_id,
                epk_uav, epk_gcs, n_uav, n_gcs, t_uav, t_gcs>
    /* GCS constructs its own signed payload */
    payload_gcs = <'GCS_confirm', pid_gcs, pid_uav, mission_id,
                epk_gcs, epk_uav, n_gcs, n_uav, t_gcs, t_uav>
    sign_gcs    = sign(payload_gcs, sk_gcs)
    msg3       = <'HandshakeVerif', pid_gcs, pid_uav, epk_gcs, epk_uav,
                n_gcs, n_uav, t_gcs, t_uav, sign_gcs>
    /* Bind msg1_stored to what GCS originally sent */
    msg1_stored = <'HandshakeInit', pid_gcs, pid_uav, n_gcs, t_gcs, epk_gcs>
    /* Shared secret computation */
    ss         = epk_uav^esk_gcs
    /* Transcript and key derivation */
    transcript  = h(<tlv(msg1_stored), tlv(msg2), tlv(msg3)>)
    salt       = h(<pid_gcs, pid_uav, transcript>)
    smk        = hkdf(ss, salt, 'SessionMasterKey', mission_id)
  in
  [ In(msg2)
  , GCS_State_1(sk_gcs, pk_gcs, pid_gcs, pid_uav, pk_uav,
            mission_id, esk_gcs, epk_gcs, n_gcs, t_gcs)
  , !GCS_TrustDB(pid_gcs, mission_id, pid_uav, pk_uav)
  ]
 --[ Eq(verify(sign_uav, payload_uav, pk_uav), true)  /* Verify UAV signature
*/
  , GCS_Sent_MSG3(pid_gcs, pid_uav, mission_id, n_gcs, n_uav, epk_gcs,
epk_uav)
  , GCS_Computed_SMK(pid_gcs, pid_uav, mission_id, smk)
  , Secret(smk)
  , CommitGCS(pid_gcs, pid_uav, mission_id, n_gcs, n_uav, epk_gcs,
epk_uav, smk)
```

```
  ]->
  [ Out(msg3)
  /* GCS erases ephemeral secret key (PFS) — esk_gcs not in any further state
*/
  ]

/*
 * STEP 4: UAV receives MSG3, verifies GCS signature, derives SMK
 * - Applies Verification Rule
 * - Verifies GCS signature → completes mutual authentication
 * - MSG1 retroactively authenticated: GCS signs EPK_GCS it originally sent
 * - Derives same SMK as GCS
 */
rule UAV_Recv_MSG3_Derive_SMK:
  let
    msg3        = <'HandshakeVerif', pid_gcs, pid_uav, epk_gcs_recv, epk_uav,
                   n_gcs, n_uav, t_gcs, t_uav, sign_gcs>
    /* Reconstruct expected GCS payload */
    payload_gcs = <'GCS_confirm', pid_gcs, pid_uav, mission_id,
                   epk_gcs_recv, epk_uav, n_gcs, n_uav, t_gcs, t_uav>
    /* Shared secret — uses EPK_GCS from MSG3 (signed by GCS) */
    ss          = epk_gcs_recv^esk_uav
    /* Transcript must use consistent values */
    msg1_reconst = <'HandshakeInit', pid_gcs, pid_uav, n_gcs, t_gcs,
epk_gcs_recv>
    msg2_reconst = <'HandshakeResp', pid_uav, pid_gcs, n_uav, t_uav,
epk_uav,
                    sign(
                      <'UAV_confirm', pid_uav, pid_gcs, mission_id,
                       epk_uav, epk_gcs_recv, n_uav, n_gcs, t_uav, t_gcs>,
                      sk_uav)>
    transcript   = h(<tlv(msg1_reconst), tlv(msg2_reconst), tlv(msg3)>)
    salt         = h(<pid_gcs, pid_uav, transcript>)
    smk          = hkdf(ss, salt, 'SessionMasterKey', mission_id)
  in
  [ In(msg3)
  , UAV_State_1(sk_uav, pk_uav, pid_uav, pid_gcs, pk_gcs,
              mission_id, esk_uav, epk_uav, epk_gcs_cached,
              n_uav, n_gcs, t_uav, t_gcs)
  ]
 --[ Eq(verify(sign_gcs, payload_gcs, pk_gcs), true)  /* Verify GCS signature
*/
  /* KEY CHECK: EPK_GCS in signed MSG3 must match what UAV cached
from MSG1 */
  , Eq(epk_gcs_recv, epk_gcs_cached)
  , UAV_Verified_MSG3(pid_uav, pid_gcs, mission_id)
  , UAV_Computed_SMK(pid_uav, pid_gcs, mission_id, smk)
```

```
    , Secret(smk)
    , CommitUAV(pid_uav, pid_gcs, mission_id, n_gcs, n_uav, epk_gcs_recv,
epk_uav, smk)
    , RunningUAV(pid_uav, pid_gcs, mission_id, n_gcs, n_uav, epk_gcs_recv,
epk_uav, smk)
    , Honest(pid_gcs)
    , Honest(pid_uav)
    ]->
    [ !SessionKey(pid_uav, pid_gcs, mission_id, smk)
    /* UAV erases ephemeral secret key (PFS) — esk_uav not in any further
state */
    ]

/* ADVERSARY MODEL */

/*
 * Long-term key compromise (for PFS analysis)
 * The adversary can compromise long-term signing keys
 */
rule Compromise_GCS_LTK:
    [ !GCS_LTK(sk_gcs, pk_gcs, pid_gcs, mission_id, serial_gcs) ]
  --[ Compromised(pid_gcs)
    , LTK_Reveal(pid_gcs)
    ]->
    [ Out(sk_gcs) ]

rule Compromise_UAV_LTK:
    [ !UAV_LTK(sk_uav, pk_uav, pid_uav, mission_id, serial_uav) ]
  --[ Compromised(pid_uav)
    , LTK_Reveal(pid_uav)
    ]->
    [ Out(sk_uav) ]

/* RESTRICTION */

restriction Equality:
    "All x y #i. Eq(x, y) @ #i ==> x = y"

restriction OnlyOnce:
    "All x #i #j. OnlyOnce(x) @ #i & OnlyOnce(x) @ #j ==> #i = #j"

/* SECURITY LEMMAS */

/*
 * LEMMA 1: Session Master Key Secrecy
 * The SMK should never be known to the adversary
 * unless a long-term key has been compromised
```

```
 */
lemma smk_secrecy:
   "All x #i.
      Secret(x) @ #i
      ==>
      ( not(Ex #j. K(x) @ #j) )
      | (Ex A #r. Compromised(A) @ #r)"

/*
 * LEMMA 2: Perfect Forward Secrecy
 * Even if long-term keys are compromised AFTER the session,
 * past session keys remain secret.
 * The adversary learns the LTK only after the SMK was derived.
 */
lemma perfect_forward_secrecy:
   "All pid_uav pid_gcs mid smk #i #j.
      UAV_Computed_SMK(pid_uav, pid_gcs, mid, smk) @ #i
     & K(smk) @ #j
      ==>
      ( Ex #r. Compromised(pid_gcs) @ #r & #r < #i )
      | ( Ex #r. Compromised(pid_uav) @ #r & #r < #i )"

/*
 * LEMMA 3: Mutual Authentication — GCS authenticates UAV
 * (Non-injective agreement)
 * If GCS commits to a session with UAV, then UAV must have
 * participated with the same parameters.
 */
lemma mutual_auth_gcs_to_uav:
   "All pid_gcs pid_uav mid n_gcs n_uav epk_gcs epk_uav smk #i.
      CommitGCS(pid_gcs, pid_uav, mid, n_gcs, n_uav, epk_gcs, epk_uav,
smk) @ #i
      ==>
      ( Ex #j. UAV_Sent_MSG2(pid_uav, pid_gcs, mid, n_uav, n_gcs, epk_uav,
epk_gcs) @ #j
        & #j < #i )
      | (Ex A #r. Compromised(A) @ #r)"

/*
 * LEMMA 4: Mutual Authentication — UAV authenticates GCS
 * (Injective agreement)
 * If UAV commits to a session with GCS, then GCS must have
 * participated with the same parameters.
 */
lemma mutual_auth_uav_to_gcs:
   "All pid_uav pid_gcs mid n_gcs n_uav epk_gcs epk_uav smk #i.
```

```
      CommitUAV(pid_uav, pid_gcs, mid, n_gcs, n_uav, epk_gcs, epk_uav,
smk) @ #i
      ==>
      ( Ex #j. GCS_Sent_MSG3(pid_gcs, pid_uav, mid, n_gcs, n_uav, epk_gcs,
epk_uav) @ #j
        & #j < #i )
      | (Ex A #r. Compromised(A) @ #r)"

/*
 * LEMMA 5: MSG1 Tampering Detection
 * If a UAV successfully completes the handshake, the EPK_GCS
 * it used must be the one GCS originally sent in MSG1.
 * This proves retroactive authentication of MSG1 via MSG3.
 */
lemma msg1_tampering_detection:
  "All pid_uav pid_gcs mid n_gcs n_uav epk_gcs epk_uav smk #i.
      CommitUAV(pid_uav, pid_gcs, mid, n_gcs, n_uav, epk_gcs, epk_uav,
smk) @ #i
      ==>
      ( Ex #j. GCS_Sent_MSG1(pid_gcs, pid_uav, mid, n_gcs, epk_gcs) @ #j
        & #j < #i )
      | (Ex A #r. Compromised(A) @ #r)"

/*
 * LEMMA 6: No Reflection Attack
 * A party should never complete a handshake with itself.
 * The asymmetric protocol labels ('UAV_confirm' vs 'GCS_confirm') prevent
this.
 */
lemma no_reflection:
  "not(Ex pid mid n_gcs n_uav epk_gcs epk_uav smk #i.
      CommitUAV(pid, pid, mid, n_gcs, n_uav, epk_gcs, epk_uav, smk) @ #i)"

/*
 * LEMMA 7: Session Key Agreement
 * Both parties derive the same SMK when in the same session.
 * Sessions are bound by matching nonces (n_gcs, n_uav).
 */
lemma key_agreement:
  "All pid_gcs pid_uav mid n_gcs n_uav epk_gcs epk_uav smk_gcs
       epk_gcs2 epk_uav2 smk_uav #i #j.
      CommitGCS(pid_gcs, pid_uav, mid, n_gcs, n_uav, epk_gcs, epk_uav,
smk_gcs) @ #i
    & CommitUAV(pid_uav, pid_gcs, mid, n_gcs, n_uav, epk_gcs2, epk_uav2,
smk_uav) @ #j
      ==>
      smk_gcs = smk_uav
```

```
      | (Ex A #r. Compromised(A) @ #r)"

/*
 * LEMMA 8: Nonce Freshness / Replay Protection
 * Each successful handshake uses unique nonces, preventing replay.
 */
lemma replay_protection:
   "All pid_uav pid_gcs mid n_gcs n_uav epk_gcs epk_uav smk
       pid_uav2 pid_gcs2 mid2 n_uav2 epk_gcs2 epk_uav2 smk2 #i #j.
      CommitUAV(pid_uav, pid_gcs, mid, n_gcs, n_uav, epk_gcs, epk_uav,
smk) @ #i
    & CommitUAV(pid_uav2, pid_gcs2, mid2, n_gcs, n_uav2, epk_gcs2,
epk_uav2, smk2) @ #j
      ==>
      #i = #j"

/*
 * LEMMA 9: Executability (Sanity Check)
 * Verifies that the protocol CAN complete successfully.
 * If this fails, the model has a structural error.
 */
lemma executability:
   exists-trace
   "Ex pid_uav pid_gcs mid smk #i #j.
      GCS_Computed_SMK(pid_gcs, pid_uav, mid, smk) @ #i
    & UAV_Computed_SMK(pid_uav, pid_gcs, mid, smk) @ #j
    & not(Ex A #r. Compromised(A) @ #r)"

end
```