Privacy-Preserving Authentication Framework for UAS Traffic Management Systems

Anas Alsoliman*, **Abdulrahman Bin Rabiah**† and **Marco Levorato***

*Donald Bren School of Information and Computer Sciences, University of California-Irvine, Irvine, CA 92697 USA
Email: aalsolim@uci.edu, levorato@uci.edu

†Marlan and Rosemary Bourns College of Engineering, University of California-Riverside, Riverside, CA 92507 USA
Email: abinr001@ucr.edu

***Abstract*—In 2015, the Federal Aviation Administration (FAA) has announced the integration of unmanned aerial vehicles (UAV) into the national airspace via a traffic management system – called UAS Traffic Management (UTM) – dedicated to Unmanned Aircraft Systems (UAS) to support advanced UAV operations such as autonomous and beyond visual line of sight (BVLOS) flight missions. The UTM incorporates an identification framework called Remote ID which mandates all UAS operators to continuously identify themselves while on flight. However, the current version of the framework lacks security features and its design has raised privacy concerns among UAS operators. This paper extends the Remote ID framework to include a Privacy-Preserving Authentication Framework that anonymously verifies the authenticity of flying UAVs. Moreover, the framework authenticates the UAV's flight permissions without revealing neither the identity of its operator nor its entire flight path, while at the same time keeping any identifying information accessible to the authorities in case of a dispute. To satisfy the proposed security and privacy requirements, a UAV's flight plan that is represented as a series of waypoints is transformed into localized UAV trajectories which create a set of contiguous flight zones, each with its own flight permission. This framework utilizes the Boneh–Gentry-Lynn–Shacham (BGLS) digital signature scheme to sign and transform each zone information into a flight permission and aggregate a set of signatures into a single signature along with additional attributes used to construct a Remote-ID message that anonymously authenticates flying UAVs.**

I. Introduction

In the last 10 years, commercial unmanned aerial vehicles (UAVs) have emerged as a key component in many applications such as goods delivery, photogrammetry, environmental research and surveying, emergency first-responders, and public safety monitoring [1–4]. However, an increased drone activity within the national airspace would eventually create various technical challenges and safety concerns. To address the latter issue, the FAA and NASA have jointly established a Research Transition Team (RTT) to initiate the development of a UAS Traffic Management (UTM) system that is separate from, but complementary to, the Air Traffic Management (ATM) system which manages manned aircrafts [5]. Under the current regulations, a UAV operator would typically need to apply for a flight authorization via an automated service called Low Altitude Authorization and Notification Capability (LAANC) [6] for piloted line of sight (LOS) flight missions only. The anticipated UTM system on the other hand will set and enforce regulations for efficient and safe autonomous and beyond visual line of sight (BVLOS) UAV operations. One of the main UTM components is the **Remote-ID** (RID) module, which will be designed to help identify UAVs in flight [7]. Importantly, the RID will also be used for detect-and-avoid applications as well as UAV-to-UAV communications.

There are some efforts in the industry attempting to implement an actual Remote-ID system. AirMap [8], Wing [9], and Kittyhawk.io [10] (each is a UAS Service Suppliers (USS) for LAANC) have demonstrated a network-based Remote-ID system [11] that exchanges UAV information across different USSs via InterUSS Platform, a platform that is designed to connect multiple USSs together [12]. On the other hand, Intel has introduced "Open Drone ID project" [13], a broadcast-based Remote-ID implementation of the "ASTM F3411 Remote ID and Tracking Specification" [14]. Open Drone ID has an app implementation for iOS and Android. It uses Bluetooth 5.0 to search for drones that are equipped with Open Drone ID for up to 1 km. The drone sends two types of broadcast messages, static messages (include static information such as drone ID) and dynamic messages (include dynamic information such as current GPS coordinates) which are broadcasted more frequently than the static ones.

The UAS Identification and Tracking (UAS ID) Aviation Rulemaking Committee (ARC) has introduced two different approaches for implementing Remote ID [15]: Direct Broadcast and Network Publishing (Figure 1). In the direct broadcast approach, the UAV continuously broadcasts identifying information such as its unique ID, tracking information, and UAS owner information. In the network publishing approach, these identifying information are uploaded to a database over the internet. Anyone with an access to such database would be able to verify the identity of any flying UAVs without the need of receiving direct broadcasts from the UAV itself.

On the 31st of December 2019, the FAA has proposed a new set of regulations for Remote Identification of Unmanned Aircraft Systems [16], which included rules defining what classes of UAVs are required to support the RID hardware and software. The document also mentioned the possibility of making the identity and the location of UAV operators publicly available which raised **privacy concerns** among the drone hobbyist community and UAS operators at large [17][18][19].

The document however did not include a definition of the process to exchange the RIDs or the structure of the RID messages. It also did not include any security requirements to preserve the integrity of the messages and prevent malicious operators from impersonating legitimate UAS operators.

Contributions: In this paper, we propose a privacy-preserving authentication framework that extends the Remote ID framework to 1) securely and anonymously authenticate a UAV during its flight without revealing the identity of its operator except to the authorities, and 2) verify the authenticity of the flight permissions held by the UAV for its current flying area and flying time without revealing the UAV's entire flight path that might lead to the UAS operator's location. To satisfy the proposed security and privacy requirements, we introduce two key contributions which we use as corner stones to build our framework: a flight plan slicing technique and a Remote-ID message structure. We can summarize our contributions as follows:

1) Flight Plan Slicing Technique: typically before each flight, a UAV is required to submit a flight request to the authorities, which includes the flight plan the UAV is intending to follow [6]. Upon request approval, the UAV receives a permission to fly in a specified region at a specified time. In case of autonomous UAV flight, a flight plan includes a flight path that is represented as a series of predetermined waypoints. Our proposed flight plan slicing technique divides the flight plan into a series of contiguous flight zones. Each zone describes a localized UAV trajectory which estimates the times at which the UAV enters and exits the zone along with the expected altitude, direction, and speed. Each flight zone is represented as timed waypoints and signed by an authoritative entity (e.g., FAA). The signed flight zone acts as a spatial and temporal *flight permission* which is used by the UAV to prove to any third party that the UAV has a permission to fly in or pass through its current region at the current time.

2) Remote-ID Message Structure: a Remote-ID (RID) is an identification message that is continuously broadcasted by the UAV. It includes a flight zone permission and a pseudonymous certificate signed by an authoritative entity. It also includes UAV telemetry information signed by the UAV. This message structure is created and exchanged by different entities in the framework via different communication channels to allow any third party authenticators to anonymously verify the authenticity of flying UAVs as well as authenticate UAVs' flight permissions without revealing neither the identity of its operators nor its entire flight path, while at the same time keeping any identifying information accessible to the authorities in case of a dispute. The message structure leverages the fast signature aggregation capability of the Boneh–Gentry-Lynn–Shacham (BGLS) digital signature scheme, which allows any arbitrary number of signatures to be aggregated as a single signature for fast and energy-efficient transmission of authentication credentials over the network. The use of signature aggregation is motivated by the fact that energy-constrained devices such as UAVs benefit from reduced size of transmitted messages more than the reduced size of messages processed locally in terms of energy consumption [20].

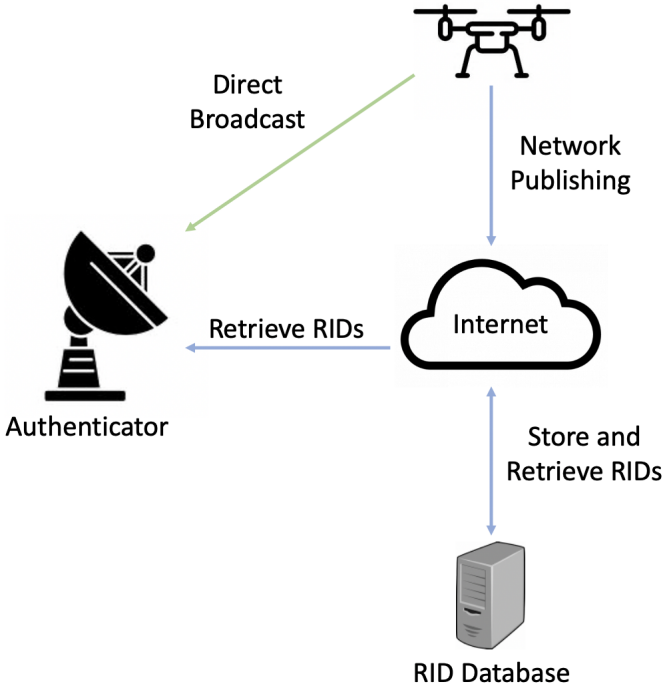

Fig. 1: Authentication Channels.

Applications: The main applications that motivated the proposed contributions are the following:

A. The base use case (which the rest of the paper will be built upon) is for UAVs to authenticate themselves to the open public while keeping their identity secured by using one flight certificate with each RID message.

B. The second use case considers a scenario where a UAV performs services for multiple clients and each client has multiple sites that the UAV needs to visit. Having multiple certificates allows the UAV to choose the permitted flight zones including all sites of a single client and revealing it to that client only, while keeping the rest of the flight path hidden.

C. In the third use case, a UAV might encounter a heavily restricted area which has a UAV detection systems at its perimeter. In this case, the UAV needs to reveal the certificates of the entire route that passes through the area to the perimeter keeper while keeping the rest of the flight path hidden.

The rest of the paper is organized as follows. In Section II, we discuss related work in the area of UAV authentication. In Section III, we provide background and a formal description of aggregate signatures. In Section IV, we discuss our flight plan slicing technique and provide the reasoning behind the proposed technique. Section V discusses the design of the privacy-preserving authentication framework and describes in detail how RID messages are constructed and disseminated among different entities in the framework. Section VI provides a security analysis of the proposed framework. In Section VII, we provide experimental results to evaluate the computational cost of BGLS as well as comparative analysis of the ECDSA and BGLS signature schemes. Section VIII concludes the paper.

II. Related Work

In [21], the authors proposed a double-authentication watermarking scheme for a network of UAVs that are clustered as nodes in a tree network architecture where cryptographically hashed watermarks are aggregated at cluster heads using a chaotic logistic mapping. Unlike our framework, the proposed scheme incurs additional overhead on the cluster head node that continuously verifies and aggregates all received data streams. Furthermore, only the nodes who possess the secret key can verify the watermarks in this scheme.

In [22], the authors propose an authentication and key agreement scheme between drones and users who connect and acquire real-time information from them. The scheme relies on secure session establishment using pseudo-identities that are facilitated by a trusted server. However, in this scheme the server must be online during the authentication phase for each session between drones and users, whereas in our proposed framework, all pseudo-identities are signed and uploaded to the drone prior to each flight mission.

In [23], the authors proposed Traceable and Privacy-Preserving Authentication scheme for UAV systems based on elliptic curve cryptography. The scheme describes a trusted authority that generates public and private key pairs for UAV manufacturers, users, and ground control stations (GCS) as certificates. Furthermore, whenever a user buys or rents a UAV, a public and private key pair is generated and provided by the UAV manufacturer as a UAV certificate. The Traceable Privacy-Preserving property comes from the fact that only the trusted authority knows the real identity of the secret key owner. The drawback of this approach is that the authority and the manufacturer are given excessive trust since they possess the secret keys of the users and drones respectively. Furthermore, the authentication process involves the UAV manufacturer in every flight request which would add an unnecessary key management overhead to the system.

In [24], the authors proposed a privacy-preserving authentication framework for UAVs to authenticate each other via mobile edge computing (MEC) using pseudonym UAV certificates that have short expiration periods (minutes to hours) to prevent UAV tracking across different MECs. Despite utilizing pseudonym certificates, a UAV's unique master public key is used each time when it connects to a MEC, which implies that a compromised MEC has the potential to revoke the anonymity of UAVs connecting to it. Furthermore, the framework does not implement any authentication procedures with external parties outside of the network nodes (UAVs and MECs) as well as it does not authenticate whether or not a UAV has a permission to fly over a MEC at the connection time.

In [25], the authors proposed a privacy preserved authentication architecture based on ID-Based Signcryption. In this architecture, UAVs are required to equip an RFID tag that contains the UAV's real ID while distributed WiFi Access Points (AP) are required to equip an RFID reader. Whenever a UAV enters an AP coverage, the UAV scans its tag via the AP's reader to send its real ID to an identity server which in turn replies back with a pseudonym ID for the UAV to use in its current AP coverage for authenticating with other UAVs under the same coverage. The main drawback of this architecture is the impractical use of RFIDs where a UAV must fly too close (6 to 9 meters) over an AP. Furthermore, the architecture assumes an AP coverage for the authentication process to take place.

III. Cryptographic Building Blocks

In this section, we review the cryptographic primitives that we utilize throughout this work. We start by describing traditional cryptographic signature schemes. We, then, provide a formal definition for aggregate signatures. Readers who are familiar with these primitives can skip this section.

Signature schemes are fundamental objects from cryptography that allow a sender to use a secret key SK to cryptographically sign a message msg and send a signature σ to a receiver. Successful validation of σ indicates that σ has been created by the sender holding SK; the security of the signature schemes makes sure that for all adversaries $\mathcal{A}$ who observes valid pairs (msg, σ), $\mathcal{A}$ cannot forge a valid σ' for any message $\mathsf{msg}' \neq \mathsf{msg}$ without knowledge of SK. For applications that require exchange of many signatures $\Sigma := \{\sigma_i\}_i$ however, such signature schemes impose additional bandwidth overhead since each $\sigma_i \in \Sigma$ has to be individually exchanged, and thus the cumulative size of signatures grows linearly with the number of signatures.

Aggregate signatures are signature schemes with features that make them attractive in many applications: they allow a set of N signatures $\Sigma := \{\sigma_i\}_i$ created by N users under N key pairs $\{(\mathsf{PK}_i, \mathsf{SK}_i)\}_i$ on N messages $\mathcal{M} := \{msg_i\}_i$, where $i \in \{1, \ldots, N\}$, to be aggregated into a single signature $\sigma = \sigma_1 \circ \cdots \circ \sigma_N$ by an aggregating entity (not necessarily associated with the users or even a trusted entity). The advantage is that the aggregate signature σ has the size of a single signature of the underlying signature scheme. We formally describe aggregate signatures next.

Definition 1: An aggregate signature scheme is a tuple of efficient algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Aggregate}, \mathsf{VerifyAggregate})$ which satisfies the following syntax, correctness, and security properties.

- **Syntax:** $\mathsf{KeyGen}(1^n)$ outputs a key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$; $\mathsf{Sign}(\mathsf{SK}_i, \mathsf{msg}_i)$ outputs a signature σ_i; $\mathsf{Verify}(\mathsf{PK}_i, \mathsf{msg}_i, \sigma_i)$ outputs a bit; $\mathsf{Aggregate}(\{\sigma_i\}_{i\in\{1,\ldots,N\}})$ outputs a signature σ; $\mathsf{VerifyAggregate}(\{\mathsf{PK}_i\}_i, \{\mathsf{msg}_i\}_i, \sigma)$ outputs a bit where $i \in \{1, \ldots, N\}$.
- **Correctness:** For all $\{(\mathsf{PK}_i, \mathsf{SK}_i)\}_i$ in the support of KeyGen, and messages $\{\mathsf{msg}_i\}_i$: $\mathsf{Verify}(\mathsf{PK}_i, \mathsf{msg}_i, \sigma_i) = 1$ holds with probability 1 where $\sigma_i = \mathsf{Sign}(\mathsf{SK}_i, \mathsf{msg}_i)$, and $\mathsf{VerifyAggregate}(\{\mathsf{PK}_i\}_i, \{\mathsf{msg}_i\}_i, \sigma) = 1$ holds with probability 1 where $\sigma = \mathsf{Aggregate}(\{\sigma_i\}_i)$ for $i \in \{1, \ldots, N\}$.

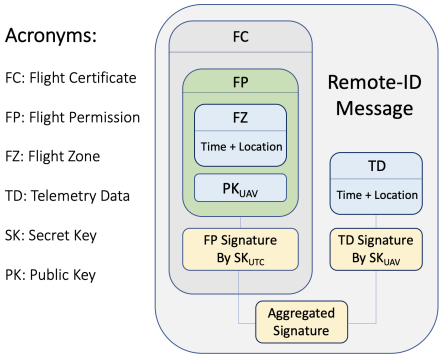

Fig. 2: Remote-ID Message Structure.

- **Security:** The probability that any efficient adversary $\mathcal{A}$ wins the aggregate signature forgery game described in [26] is negligible.

Remark. In the aggregate signature security game described in [26], $\mathcal{A}$ is given a lot of power, namely it holds $(\mathsf{PK}_1), (\mathsf{SK}_2, \mathsf{PK}_2), \ldots, (\mathsf{SK}_N, \mathsf{PK}_N)$, chooses messages $\mathcal{M} := \{\mathsf{msg}_1, \ldots, \mathsf{msg}_N\}$ to be signed, has access to a signing oracle for SK_1 (without $\mathcal{A}$ knowledge of SK_1) to sign any $\mathsf{msg}' \neq \mathsf{msg}_1$ and is challenged by a challenger $\mathcal{C}$ to produce the aggregate signature $\sigma = \mathsf{Aggregate}(\{\sigma_i\}_i)$ where $i \in \{1, \ldots, N\}$, indicating inability of $\mathcal{A}$ to forge an aggregate signature σ; please see [26] for details on security.

In our authentication framework, we use the provably secure aggregate signature scheme BGLS proposed in [26]. Our framework design is modular, so that if future, more efficient, aggregate signature schemes are proposed, we could flexibly use them instead of BGLS.

IV. UAV Flight Planning & Flight Plan Slicing

In autonomous flight missions, an Unmanned Aerial System (UAS) operator generates a flight plan for the UAV to follow. There are different techniques and algorithms for different objectives and motivations to calculate and design a UAV flight plan. Generally, when a flight planning algorithm calculates the optimal path based on the task's navigational requirements, it outputs the path as a series of waypoints; a 3D point in space that includes the latitude, longitude, and altitude of the waypoint. In this framework, our main objective is to anonymously authenticate the flight permissions of UAVs without revealing their entire flight path while asserting to any outside authenticator that the UAV possesses a permission to fly over a particular area.

In this section, we introduce a Flight Plan Slicing technique; an approach for dividing the waypoints that is being followed by a UAV into smaller blocks of contiguous Flight Zones (FZs). Each FZ is a localized UAV trajectory that estimates the time at which the UAV enters and exists the FZ given the speed of the UAV and the path it follows. Therefore, each FZ can be transformed into a signed Flight Certificate (FC) which can be used to prove to any third party that the UAV has a permission to fly over its current time and location (Figure 2). To introduce anonymity between different FZs, each FC includes a unique cryptographic public key (PK) that belongs to the UAV. Therefore, the UAV would appear as a new UAV whenever it enters and exits a new FZ. A more detailed discussion about the structure of Remote-ID message (Figure 2) will be explored in section V. In this section, we discuss the basis of determining the size of each FZ. In our discussion, we consider Linear Flight Planning; a flight planning technique for UAVs traveling mostly in straight lines. Linear Flight Planning is a popular flight planning technique since most UAV flight missions prefer the shortest straight path between two waypoints. Furthermore, more complex flight plans can be constructed from a composition of linear flight plans.

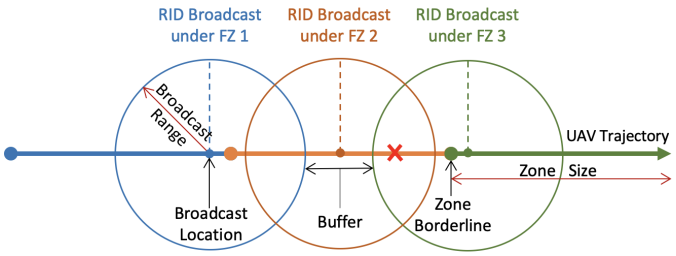

Fig. 3: Three (color-codded) Flight Zones describe a worst-case scenario where a UAV performed a broadcast right before exiting FZ1 and right after entering FZ3. The buffer range prevented broadcasts from FZ1 and FZ3 to overlap. Authenticator at the Red X can only correlate a maximum of two zones (FZ2 and FZ3) to the same UAV.

Generally, having many FZs in a flight plan makes it harder for authenticators to successfully correlate the PKs to the same UAV by collecting all (or most of) the FZs from Remote-ID (RID) broadcasts. In contrast, if we have very few FZs such as two FZs for the entire flight path, an authenticator located near or right at the borderline separating the two FZs would be in the broadcast coverage of both FZs (Figure 3). Therefore, when the authenticator receives the RID broadcasts (which includes the FC) from the UAV, it can deduce that the RIDs received from FZ_i and FZ_{i+1} belong to the same UAV even if the PKs are not correlated. In another scenario, if we have three FZs for the entire flight path and two authenticators are located near the borderlines separating the FZs, each authenticator would possess part of the entire flight path by listening to RID broadcasts, and if the two authenticators collaborate with each other, they both can deduce that the collected RIDs belong to the same UAV. It is worth noting that even if an authenticator was able to reconstruct the entire flight path of a UAV, the authenticator cannot conclude for certain that the reconstructed path is the complete path of the UAV. This also implies that successfully correlating some of the FZs to the same UAV would partially compromise the UAV's anonymity and the more FZs are correlated the bigger the compromise would get.

Our main approach to tackle this issue is to first determine the UAV's broadcast coverage area. Each wireless protocol and hardware has a minimum acceptable signal strength. For example, typical WiFi adapters use omnidirectional antennas

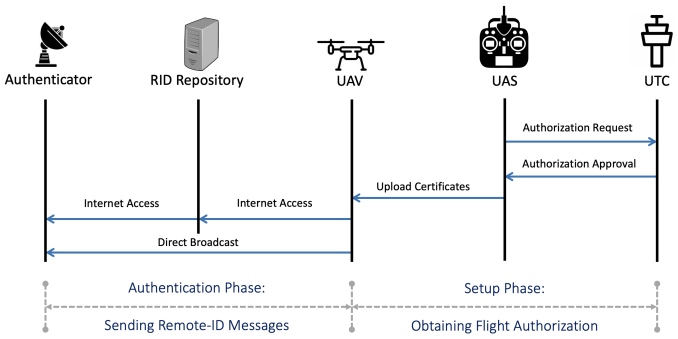

Fig. 4: Framework Workflow.

and require at least -80 dBm signal strength. Let's denote the minimum acceptable signal strength threshold as θ. Next we use the Link Budget equation $Receivedpower(dB) = transmittedpower(dB) + \sum gains(dB) - \sum losses(dB)$ to estimate θ, which can be written as: $P_{RX} = P_{TX} + G_{RX} + G_{TX} - L_{RX} - L_{TX} - L_{LM} - L_{FS}$ where P_{RX} is the authenticator's received power which is also the threshold θ, P_{TX} is transmitter output power, G_{RX} is receiver antenna gain, G_{TX} is transmitter antenna gain, L_{RX} is receiver losses, L_{TX} is the transmitter losses, and L_{FS} is the free space loss. Furthermore, L_{FS} is expressed as: $L_{FS} = 20\log_{10}(\frac{4\pi d}{\lambda})$ where λ is the wavelength and d is the distance in the same unit as the wavelength. Then, we use the Link Budget function to solve for d such that: $d = \frac{\lambda \sqrt[20]{10^{P_{TX}}+10^{G_{RX}}+10^{G_{TX}}-10^{L_{RX}}-10^{L_{TX}}-10^{L_{LM}}-10^{P_{RX}}}}{4\pi}$ Once we solve for d, we can use the solution to estimate the broadcast coverage radius r given the UAV's altitude alt such that: $r = \sqrt{d^2 - alt^2}$.

After finding r, we can now define the size of each FZ to be at least bigger than the coverage diameter $2r$. In real-life scenarios, it is inevitable that an authenticator might be located between two adjacent FZs where the authenticator can receive two RIDs from the same UAV that happened to pass over two adjacent FZs. Therefore, setting the size of each FZ to $2r$ ensures that no authenticator can receive more than two RIDs of adjacent FZs from the same UAV. Furthermore, to ensure that broadcasts from neighboring FZs do not overlap on a 3rd FZ and to defend against unexpected high G_{RX}, a buffer b is added to the FZ size such that $FZSize = 2r + b$ where $b \leq \frac{r}{2}$ (Figure 3).

After defining the size of each FZ, the entire flight path can now be divided into FZs prior to the flight mission where each two waypoints that have a distance longer than $2r+b$ is divided by FZ size and any series of waypoints that is shorter than $2r + b$ are combined together until it reaches the acceptable FZ size. For each FZ, a UAV's entry and exit time to the FZ is estimated based on the takeoff time, the distance to the FZ, and the expected UAV's speed.

V. Authentication Framework Design

The authentication framework defines the process of constructing and exchanging the Remote-ID (RID) messages between different entities in the framework. To recap, an RID is a message that is continually broadcasted by a UAV to anonymously authenticate itself to any third party authenticators as well as to prove to the authenticator that the UAV has a permission from an authoritative entity to fly over its current flying location at the time of the broadcast.

There are five participants in the authentication framework (Figure 4), Unmanned Aerial Vehicle (UAV), Unmanned Aerial System (UAS), UAS Traffic Controller (UTC), UAS Service Supplier (USS), and an Authenticator. The UAV is the entity that is required to anonymously and continually authenticate itself by broadcasting/uploading its RID messages. The UAS is the operator of the UAVs (e.g. a UAV hobbyist or a commercial drone delivery company). A single UAS operator may have one or multiple UAVs simultaneously. UTC is the authoritative entity that regulates and oversees the UAV operations over an airspace (e.g. FAA). The UTC is the equivalent to Air Traffic Controller (ATC) that manages manned aircraft operations. Similar to the current regulations [6], all UAVs participating in the authentication framework are required to obtain a digitally signed flight authorization from a UTC prior to each flight mission. The USS is an entity that provides services to other entities in the framework to support various UTM operations (e.g. an online database that stores all UAVs digital certificates and RIDs). In our framework, an RID repository is required to support the anonymous authentication process via the internet (Figure 4). A public authenticator is any entity that receives UAV RIDs (either via direct broadcast from the UAV or over the internet from an RID repository) to anonymously authenticate the UAV based on the RID credentials.

Framework Overview

The framework process is executed in two different phases: a Setup Phase and an Authentication Phase (Figure 4). In the setup phase, a UAS operator submits a flight authorization request to the overseeing UTC before each flight mission and obtains an authorization approval that contains a list of Flight Certificates (FCs), where each FC is used to authenticate the UAV over a specific region within a specific time period. In the authentication phase, the UAV uses the appropriate FC to construct an RID message and continually disseminates the RIDs over two different wireless channels, a direct RID broadcast or uploading RIDs to an RID repository over the internet (Figure 1).

Every RID includes an FC which is a spatiotemporal flight permission, a flight permission bounded by a geographic location and a time window. Furthermore, each FC has its own PK which makes every FC acting as a pseudonymous digital certificate for the UAV that certifies its PK and its flight permissions for the current fly zone and time. Since every FC has a different PK, an authenticator cannot track a specific UAV to its destination or back to its takeoff site without physically following the UAV since the UAV would use a new PK for each flight zone and would appear as a new UAV whenever exiting or entering a new flight zone. Furthermore, since the list of Flight Certificates (FCs) form

a series of contiguous flight zones that are bounded by a sequential arrival and departure times, a UAV cannot deviate from its designated flight path or fly at a different times other than the time that is approved by the UTC without risking a violation detection by authenticators.

A. Setup Phase:

Authorization Request: Prior to each flight, a UAS operator must obtain a flight authorization from the UTC which will be used to construct each RID message (Figure 2) during the UAV's flight time.

1) First, the UAS operator creates a flight plan for the UAV assigned to the mission and divides it into flight zones $FZ_i \in \{FZ_1, FZ_2, ..., FZ_n\}$ where $i \in \{1, 2, ..., n\}$ and each FZ_i is the expected UAV trajectory over a specific region within a time period as discussed in section IV.
2) For each FZ_i, the operator generates a public/private key pair $(PK_i, SK_i) \in \{(PK_1, SK_1), (PK_2, SK_2), ..., (PK_n, SK_n)\}$ and appends each PK_i to a FZ_i as a pair to create a "flight permission" $FP_i = (PK_i, FZ_i)$.
3) The set "Flight Plan Vector" *FPV* is defined as *FPV* = $\{FP_1, FP_2, ..., FP_n\}$ where $FP_i = (PK_i, FZ_i)$.
4) Then the operator creates an authorization request *Req* that includes the following attributes: `UAS ID`, `UAV ID`, `FPV`, and `UAS public key` PK_{UAS}, and signs the request with the operator's private key SK_{UAS} such that $\sigma_{Req} := Sign(SK_{UAS}, Req)$.
5) Next, the UAS operator sends (Req, σ_{Req}) to the UTC over a secure channel such as TLS. The UTC then verifies the UAS request signature such that $Verify(PK_{UAS}, Req, \sigma_{Req}) = 1$.

Authorization Approval: Once the signature is verified and the flight request gets approved, the UTC creates an authorization approval as follows:

1) The UTC signs each $FP_i \in FPV$ using UTC's private key SK_{UTC} to create FP_i signature σ_{FP_i} such that $\sigma_{FP_i} := Sign(SK_{UTC}, FP_i)$. Each signed FP_i is a flight certificate $FC_i := (FP_i, \sigma_{FP_i})$ that is used by the UAV to authenticate itself over a flight zone within a specific period of time.
2) The UTC, then, populates a "Flight Certificates Vector" set *FCV* which is defined as $FCV = \{FC_1, FC_2, ..., FC_n\}$ where $FC_i := (FP_i, \sigma_{FP_i})$
3) The UTC sends the *FCV* back to the UAS as an authorization approval over a secure channel such as TLS.
4) Once the UAS receives the *FCV* from the UTC, the UAS groups all the secret keys SK_{UAV_i} generated at the Authorization Request step as a "Secret Key Vector" *SKV* where $SKV = \{SK_{UAV_1}, SK_{UAV_2}, ..., SK_{UAV_n}\}$ and uploads the *FCV* and *SKV* to the UAV assigned to the mission.

B. Authentication Phase:

During flight time, the UAV is required to authenticate itself by continually sending out RID messages. Each RID message can be communicated via two different channels: by direct broadcast and through the internet (Figure 1). Authenticators within the the UAV's broadcast range can receive and decode the RIDs. Authenticators with an active internet connection can retrieve the RIDs from an RID repository over the internet. When the UAV flies over a region, the authentication process takes place as follows:

Signing the RID:

1) The UAV selects the FC_i from FCV that has an FZ_i matching its current flight area at its current flight time.
2) Then the UAV records its current telemetry data TD that includes the current time and GPS location readings.
3) After that, the UAV creates and signs an RID message using the secret key SK_{UAV_i} matching the PK_i in the currently used FC_i such that $\sigma_{RID} := Sign(SK_{UAV_i}, RID)$ where $RID = (TD, FC_i)$.
4) To reduce the message size, the UAV uses a signature aggregation function to aggregate the UTC signature σ_{FP_i} with the UAV signature σ_{RID} to get a new σ_{RID} such that $\sigma_{RID} := \mathsf{Aggregate}(\sigma_{FP_i}, \sigma_{RID})$.
5) Finally, the UAV sends out the tuple (RID, σ_{RID}) via a direct broadcast and to the internet.

Verifying the RID:

When an authenticator receives an RID either via a direct broadcast or from the internet, it performs the following:

1) The authenticator takes the aggregate signature σ_{RID}, PK_{UAV_i}, and PK_{UTC} as inputs for an aggregate signature verification function such as $\mathsf{VerifyAggregate}(\{PK_{UAVi}, PK_{UTC}\}, \{FP_i, TD\}, \sigma_{RID})$.
2) If the aggregate signature verifies successfully, the authenticator cross-references the time and location of FZ_i in FC_i with its own current time and location. If the authenticator is within range of FZ_i and the time of receiving the RID is within FZ_i time window, the RID is accepted. Otherwise, appropriate actions are taken for a suspected spoofing attempt. Note that the authenticator did not cross-reference with TD because the UAV cannot be fully trusted since it can forge its own telemetry readings. The main purpose of TD is to provide non-repudiation and holds the UAV accountable when a forged TD is detected. Another purpose of the TD is when the actual time and location of the UAV is required such as UAV-to-UAV communications. A more thorough security analysis is discussed in the next section, section VI.

VI. Security Analysis

In this section, we analyze the security of the proposed framework and the RID structure with respect to different attack scenarios.

Impersonation Attack: Suppose a rogue UAV attempts to enter a restricted area but does not have permissions from the UAS Traffic Controller (UTC) to do so. The rogue UAV might attempt to forge its own flight permissions and use it to construct its own RID messages, but since the UAV does not

TABLE I: BGLS and ECDSA Comparison.

Signature Scheme	Key Generation (milliseconds)	Message Signing (milliseconds)	Public Key Size (bits)	Signature Size (bits)
BGLS	0.25	4.45	768	384
ECDSA	0.714	0.76	512	512

have Flight Certificates (FCs) legitimately signed by the UTC, the RIDs will not be validated by the authenticators.

Replay Attack: A rogue UAV might attempt to intercept a freshly broadcasted RID that are constructed by a legitimate UAV and re-broadcast (replay) it at a different time and/or at a different location to gain access to a restricted area. Each RID includes a FC that is bounded by a specific time and a specific location. Therefore when an authenticator receives an RID with a FC that does not match its current time and location, the RID will be rejected.

Misbehaving UAVs: Although only registered and certified Unmanned Aerial System (UAS) operators are allowed to submit flight requests by using their own certificates to obtain flight approvals for their UAVs, there is still the possibility of having misbehaving UAVs in the network that are hijacked, malfunctioned, or maliciously altered by their UAS operators to deviate from its designated paths. We remark that in the proposed scheme, each UAV appends its Telemetry Data (TD) to every RID message which includes current time and location of the RID broadcast. Therefore, if a misbehaving UAV attempts to change the course of its designated flight path, its telemetry data will not match the spatiotemporal bounds of the accompanying FC. Furthermore, if the UAV attempts to forge its own TD to falsely match the bounds of FC, it will not match any authenticator's current time and location. It is worth noting that the main purpose for including TD is to follow the FAA's guidelines for safe navigation when interacting with other participants in the UAS Traffic Management (UTM) system.

Leaked Flight Certificate Vector (FCV): FCV is a list of Flight Certificates (FCs) that includes the signed path for the UAV to follow. If the FCV (or part of it) lost its confidentiality before the start time of the leaked FCs, any forged RIDs by rogue UAVs using the leaked FCs will not be accepted by authenticators since for each RID to be accepted, it must be signed by the owner of the Secret Key SK_{UAV_i} which belongs to the public key PK_{UAV_i} that is embedded inside each FC_i for each FC period i. The list of all secret keys $\{SK_1, ..., SK_n\}$ are generated by the controlling UAS and are never sent over the network.

Anonymity and Untraceability: Each FC_i in FCV is individually signed by the UTC and includes a unique PK_{UAV_i}. Each PK_{UAV_i} is randomly generated and is not tied to any identifying information which implies that authenticators cannot deduce the owner of the UAV from its RIDs. The mapping to each PK_{UAV_i} with the actual identity of the UAV is traced via the public key of the UAV operator PK_{UAS} which is only accessible to the UTC since it is the entity that signed all PK_{UAV_i} at the setup phase in section V. Furthermore, authenticators cannot pinpoint from where the UAV is coming from or where the UAV is going to since the FC_i in the RID is only exposing a limited range of where the UAV is allowed to operate in.

VII. Performance Evaluation

In this section, we provide a performance evaluation of the signature aggregation scheme using Mariano Sorgente's implementation of Boneh–Gentry–Lynn–Shacham (BGLS) signature scheme [27] as well as a comparative analysis with Brian Warner's implementation of Elliptic Curve Digital Signature Algorithm (ECDSA) signature scheme [28]. Since our framework focuses on the performance on the UAV side, we evaluate the message sizes of both BGLS and ECDSA as well as the processing time of key generation and the message signing operation of both signature schemes. We also evaluate the processing time of signature aggregation using BGLS.

Message Sizes:

Table I shows the size of the signature and the public key of both schemes. At first, it might appear that the total size of ECDSA (Signature + Key = 1024 bit) is less than BGLS (Signature + Key = 1152 bit), but each RID message requires at least two signatures and up to the entire FC_i signatures. Therefore, the minimum size of an ECDSA is 1536 bits with additional 512 bits for each additional signature, while BGLS will always have a fixed cryptographic size which is 1152 bit regardless of the number of signatures. In energy-constrained devices such as UAVs, it is critical to reduce the size of messages to be communicated since the energy consumption of sending a single byte over the network is higher than the energy consumption of processing a single byte locally [20].

Computational Costs of Key Generation & Message Signing: We have generated 10,000 key pairs and signatures for each scheme. It took BGLS 2.5 seconds to generate 10,000 public and private keys which means 0.25 ms for each key generation operation (Table I). On the other hand, it took ECDSA 7.14 seconds to generate the same number of keys, that is 0.714 ms per key pair. This shows that BGLS provides an efficient key generation algorithm and having a key pair for each Flight Certificate is not a computational bottleneck.

Regarding message signing, BGLS took 44.5 seconds to sign 10,000 distinct messages (4.45 ms per signature) while it took ECDSA 7.6 seconds to sign the same set of messages (0.76 ms per signature). Even though ECDSA's signing operation is more efficient than BGLS's, our framework requires a UAV to perform a single signing operation with every broadcast, and 4.45 ms per broadcast is not a prohibitive performance since the broadcasting frequency in realistic

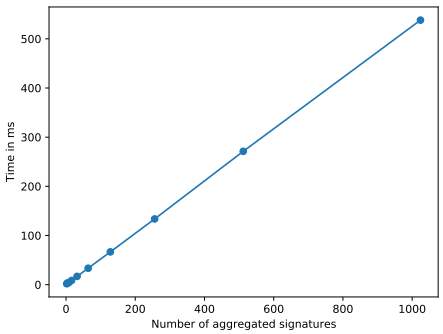

Fig. 5: Signature Aggregation.

scenarios is expected to be reasonably low (e.g. five broadcasts per second).

Computational Costs of Signature Aggregation:

Figure 5 shows the efficiency of signature aggregation. Aggregating the minimum number of signatures required by the framework (i.e. two signatures) takes 1.8 ms. Overall, the figure shows a linear relationship between the number of aggregated signatures and the computational time. In our framework, most aggregations that require more than two signatures can be computed offline since the UAV's flight plan will most likely include the special areas where the UAV is expected to reveal more than one Flight Certificate (FC) at the same time.

VIII. Conclusions

In this paper, we proposed a Privacy-Preserving Authentication Framework for UTM systems based on the FAA's Remote-ID module. The main objective of the framework is to anonymously authenticate flying UAVs and at the same time verify their flight permission without exposing their entire flight route. To achieve authentication anonymity, a flight plan slicing technique is introduced where the flight plan of a UAV is divided into blocks of flight zones that are bounded by a time and a location while each zone has its own public key. A unique structure of a Remote-ID message is also proposed where a UAV carries multiple signatures and uses signature aggregation algorithm to send all required signatures as one signatures. A security analysis and a performance evaluation is also introduced.

References

[1] Meduim. *Wing Launches America's First Commercial Drone Delivery Service to Homes in Christiansburg, Virginia*. https://bit.ly/3n2wGJ5. 2019 (accessed May 5, 2020).

[2] Lora Kolodny. *Zipline, which delivers lifesaving medical supplies by drone, now valued at $1.2 billion*. https://cnb.cx/2EKnGHa. 2019 (accessed May 5, 2020).

[3] RBR Staff. *Airobotics Creates In-House Drone Payload for Inspection, Safety Uses*. https://bit.ly/3i6yvAU. 2019 (accessed May 5, 2020).

[4] Jillian D'Onfro. *Amazon's New Delivery Drone Will Start Shipping Packages 'In A Matter Of Months'*. https://www.forbes.com/sites/jilliandonfro/2019/06/05/amazon-new-delivery-drone-remars-warehouse-robots-alexa-prediction/#3bce6a4145f3. 2019 (accessed May 5, 2020).

[5] FAA. *Unmanned Aircraft System Traffic Management (UTM)'*. https://www.faa.gov/uas/research_development/traffic_management/. 2020 (accessed May 5, 2020).

[6] FAA. *UAS Data Exchange (LAANC)*. https://www.faa.gov/uas/programs_partnerships/data_exchange/. 2020 (accessed May 5, 2020).

[7] FAA. *UAS Remote Identification*. https://www.faa.gov/uas/research_development/remote_id/. 2020 (accessed May 5, 2020).

[8] AirMap. *Deploy Digital Airspace Automation For UAS Traffic Management*. https://www.airmap.com/authorities/. 2020 (accessed July 5, 2020).

[9] Wing. *Wing delivery is easy to use*. https://wing.com/how-it-works/. 2020 (accessed July 5, 2020).

[10] Kittyhawk. *Remote ID & Commercial Drones*. https://kittyhawk.io/remote-id/. 2020 (accessed July 5, 2020).

[11] AirMap. *AirMap, Wing, and Kittyhawk.io Demonstrate InterUSS Network-Based Remote ID Application*. https://www.airmap.com/airmap-wing-kittyhawkio-demonstrate-network-based-remote-identification-interuss-platform/. 2020 (accessed July 5, 2020).

[12] InterUSS. *The InterUSS Project enables trusted, secure and scalable interoperability between UAS Service Suppliers (USSs) to further safe, equitable and efficient drone operations*. https://interussplatform.org/. 2020 (accessed July 5, 2020).

[13] OpenDroneID. *Welcome to opendroneid.org*. https://www.opendroneid.org/. 2020 (accessed July 5, 2020).

[14] ASTM. *Standard Specification for Remote ID and Tracking*. https://www.astm.org/Standards/F3411.htm. 2020 (accessed July 5, 2020).

[15] FAA. *Advisory and Rulemaking Committees - UAS Identification and Tracking ARC*. https://www.faa.gov/regulations_policies/rulemaking/committees/documents/index.cfm/document/information/documentID/3302. 2017 (accessed May 5, 2020).

[16] FAA. *Remote Identification of Unmanned Aircraft Systems*. https://www.regulations.gov/docket?D=FAA-2019-1100. 2020 (accessed May 5, 2020).

[17] Jon Hegranes. *FAA's proposed remote ID rules should make compliance easy*. https://techcrunch.com/2020/02/12/faas-proposed-remote-id-rules-should-make-compliance-easy/. 2020 (accessed May 5, 2020).

[18] Miriam McNabb. *Chris Korody Speaks Out on Remote ID for Drones: "A Billion Dollar Solution to a Non-Existent Problem"*. https://dronelife.com/2020/02/26/chris-karody-speaks-out-on-remote-id-for-drones-a-billion-dollar-solution-to-a-non-existent-problem/. 2020 (accessed May 5, 2020).

[19] MARCUS CHAVERS. *It's Privacy, Stupid. FAA Drone Remote ID Rule Comments Open Tomorrow*. https://www.newsledge.com/its-privacy-stupid-faa-drone-remote-id-rule/. 2020 (accessed May 5, 2020).

[20] L. Casado and P. Tsigas. "ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System". In: *NordSec*. Springer, 2009.

[21] Jianguo Sun et al. "A data authentication scheme for UAV ad hoc network communication". In: *The Journal of Supercomputing* (2017), pp. 1–16.

[22] Mohammad Wazid et al. "Design and analysis of secure lightweight remote user authentication and key agreement scheme in Internet of drones deployment". In: *IEEE Internet of Things Journal* 6.2 (2018), pp. 3572–3584.

[23] Chin-Ling Chen et al. "A traceable and privacy-preserving authentication for UAV communication control system". In: *Electronics* 9.1 (2020), p. 62.

[24] Yifan Tian, Jiawei Yuan, and Houbing Song. "Efficient privacy-preserving authentication framework for edge-assisted Internet of Drones". In: *Journal of Information Security and Applications* 48 (2019), p. 102354.

[25] Sana Benzarti, Bayrem Triki, and Ouajdi Korbaa. "Privacy Preservation and Drone Authentication Using ID-Based Signcryption." In: *SoMeT*. 2018, pp. 226–239.

[26] Dan Boneh et al. "Aggregate and verifiably encrypted signatures from bilinear maps". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2003, pp. 416–432.

[27] Mariano Sorgente. *Chia-Network/bls-signatures*. https://github.com/Chia-Network/bls-signatures/tree/master/python-bindings. 2020 (accessed July 5, 2020).

[28] Brian Warner. *python-ecdsa*. https://github.com/warner/python-ecdsa. 2020 (accessed July 5, 2020).