Lightweight Privacy-Preserving Proximity Discovery for Remotely-Controlled Drones

Pietro Tedeschi
pietro.tedeschi@tii.ae
Technology Innovation Institute
Abu Dhabi, United Arab Emirates

Savio Sciancalepore
s.sciancalepore@tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

Roberto Di Pietro
roberto.dipietro@kaust.edu.sa
King Abdullah University of Science and Technology - CEMSE - RC3
Thuwal, Saudi Arabia

ABSTRACT

Discovering mutual proximity and avoiding collisions is one of the most critical services needed by the next generation of Unmanned Aerial Vehicles (UAVs). However, currently available solutions either rely on sharing mutual locations, neglecting the location privacy of involved parties, or are applicable for fully autonomous vehicles only—leaving unaddressed Remotely-Piloted UAVs' safety needs. Alternatively, proximity can be discovered by adding sensing capabilities. However, in addition to the cost of the sensors, the complexity of integration, and the toll on the energy budget, the effectiveness of such solutions is usually limited by short detection ranges, making them hardly useful in high-mobility scenarios.
In this paper, we propose LPPD (an acronym for Lightweight Privacy-preserving Proximity Discovery), a unique solution for privacy-preserving proximity discovery among remotely piloted UAVs based on the exchange of wireless messages. LPPD integrates two main building blocks: (i) a custom space tessellation technique based on randomized spheres; and, (ii) a lightweight cryptographic primitive for private-set intersection. Another feature enjoyed by LPPD is that it does not require online third parties. LPPD is rooted in sound theoretical results and is supported by an experimental assessment performed on a real drone. In particular, experimental results show that LPPD achieves 100% proximity discovery while taking only 39.66 milliseconds in the most lightweight configuration and draining only the $5 \cdot 10^{-6}$% of the UAV's battery capacity. In addition, LPPD's security properties are formally verified.

ACM Reference Format:
Pietro Tedeschi, Savio Sciancalepore, and Roberto Di Pietro. 2023. Lightweight Privacy-Preserving Proximity Discovery for Remotely-Controlled Drones. In *Annual Computer Security Applications Conference (ACSAC '23), December 4–8, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3627106.3627174

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs), often referred to as *drones*, are today widespread in the industry and commercial domains, thanks to enhanced mobility, autonomy, and flexibility features [28]. Applications based on the use of UAVs can be found today in several heterogeneous fields, including Health, Delivery, Transportation, Industrial, and Military use-cases, to name a few [26].

ACSAC '23, December 4–8, 2023, Austin, TX, USA

According to recent forecasts by leading specialized companies, drone deliveries accounted for 210,000 units in 2020, and they are expected to more than double by 2023 [33]. Also, leading goods delivery companies such as Amazon are already offering services for automated goods transportation via drones (e.g., Amazon Prime Air [12]), paving the way for a future where hundreds of UAVs will be flying at the same time, to perform different automatic tasks. In this context, UAVs should integrate effective solutions for proximity discovery and collision avoidance to safeguard the safety of flying devices, goods, and especially people [38] [14].

Many solutions have been proposed in the last decade for proximity discovery and collision avoidance on UAVs (see Sec. 6 for a detailed overview). However, besides being effective only in scenarios with specific constraints, very few currently available solutions entirely took location privacy into account. Indeed, when moving according to a particular mission, the source and destination locations of the UAVs are private. The leakage of previously-cited information might lead to the discovery of sensitive details, such as the location of the storage centers of the delivery company or the association between specific goods and customers. In addition, the delivery of real-time location information (as recommended by the recent *Remote ID* specification [11]) to potentially untrusted entities might also lead to the capture of the UAVs, leading to relevant economic losses [29], [35].

In this context, most solutions that allow detecting approaching collisions without sharing UAVs locations require dedicated sensors. Moreover, these sensors usually work only in a very short range around the devices and might be ineffective when vehicles move at a significant speed. To tackle these issues, some recent proposals, i.e., the ones by the authors in [30] and [25], provided solutions for fully autonomous vehicles based on the precise knowledge of future trajectory points (timestamps and locations). However, such solutions suffer from two main drawbacks: (i) they require several over-the-air messages—hence taking a toll on the energy budget—; and (ii) they cannot be applied for Remotely-Piloted Aircraft Systems (RPASs), where remote pilots issue unplanned commands for movements to the UAV at run-time (see Sec. 6).

Thus, the existing literature currently misses an effective privacy-preserving proximity discovery solution for remotely-piloted UAVs, not relying on plain-text location sharing and not requiring knowledge of the destination of the mission in advance, but at the same time also being reliable and efficient in scenarios characterized by high dynamicity and mobility.

Contribution. In this paper, we propose LPPD (an acronym for Lightweight Privacy-preserving Proximity Discovery), a privacy-preserving protocol for proximity discovery on remotely-piloted

UAVs. LPPD allows two remotely-piloted UAVs in direct wireless visibility to exchange messages and identify, in a very short time, if they are at risk of collision. To this aim, LPPD customizes and applies a solution for private-set intersection in the context of location privacy, extending such a technique with a novel space tessellation logic suitable for proximity detection. Overall, LPPD allows the two UAVs to privately discover mutual proximity to let them agree on different safe paths and protect the safety of involved devices and people.

We tested the performance of LPPD via extensive simulations and real experimentation on a prototype running on the popular 3DR-solo drone [5]. Our results indicate that LPPD guarantees the detection of all possible collisions at run time while requiring negligible time and energy overhead on commercial UAVs. For instance, when configured to work with the minimum acceptable security level of 80 bits, LPPD takes only 39.66 ms to complete, draining just ≈ 14.15 mJ of energy, i.e., the $5 \cdot 10^{-6}$ % of the UAV battery. LPPD can also achieve stronger security guarantees at the cost of a little increase in the incurred overhead.

Overall, LPPD emerges as a customizable, flexible, and effective solution for privacy-preserving proximity detection. With reference to the latest Remote ID regulation, LPPD makes a step forward, providing not only remote identification but also proximity detection and added location privacy. Thus, LPPD could be the ideal solution in contexts where a Remote ID-like regulation is to be adopted, e.g., in military and tactical use-cases, or as further advancement of the Remote ID rules itself, with LPPD merging the best of both worlds: accountability and privacy.

Roadmap. The paper is organized as follows. Sec. 2 introduces the scenario and adversarial model, Sec. 3 provides the details of LPPD, Sec. 4 discusses the security features of LPPD, Sec. 5 provides an extensive performance assessment of LPPD via simulations and experiments on a real UAV, while Sec. 6 discusses the related work and highlights the advantages of LPPD compared to the current state of the art. Finally, Sec. 7 tightens conclusions and draws future work.

2 SCENARIO AND ADVERSARY MODEL

This section introduces our scenario (Sec. 2.1), adversary and threat models (Sec. 2.2).

2.1 Scenario

We consider N UAVs, namely, $u_1, u_2, \ldots, u_N$, distributed in a given geographical area, as depicted in Fig. 1.

We assume that the UAVs are RPASs, piloted remotely via either an RF controller or commands issued at a ground control station and delivered to the UAV via a mobile cellular connection. The specific way the UAVs are piloted is out of the scope of this contribution.

Each UAV features a Global Navigation Satellite System (GNSS) module, such as a Global Positioning System (GPS) receiver, which allows it to receive geo-location data from dedicated satellites to estimate its location, with a maximum accuracy of δ meters. Such inaccuracy might be different based on the environment where the drone is located, e.g., higher in urban areas than in open rural areas. Based on the current location, the drone might calibrate the value of δ accordingly. Overall, LPPD assumes the availability of a stable GPS location, i.e., we assume that, at run time, the current almanac, initial position, Ephemeris and time information of the GPS are all set. We acknowledge that the time-to-first-fix might be significant, but this is usually achieved at the turn-on of the drone. At the same time, such information is already available when the drone is accomplishing its intended mission.

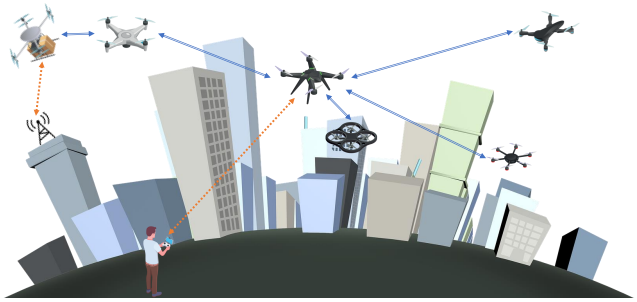

Figure 1: Reference Scenario.

We also assume that each UAV is equipped with a wireless communication technology, e.g., Wi-Fi, allowing communication on one of the Wi-Fi channels available in the unlicensed frequency band $B = [2.4 - 2.5]$ GHz, to communicate with other UAVs in the same area. Note that this is a standard feature, not only for UAVs but also for commercial Wi-Fi access points, known often with the name of *Wi-Fi Direct* [18].

We assume each UAV would like to detect other vehicles nearby to avoid physical collisions using a dedicated network protocol. Moreover, we assume that any UAV is unwilling to disclose its actual and future locations, as they are private data that could reveal sensitive information. Indeed, if exposed, such information can be used for several malicious activities (see Sec. 2.2).

Without loss of generality, we assume that the UAVs does not feature a persistent Internet connection. Although many UAVs can integrate hardware modules to provide continuous Internet connectivity, such a connection might not always be available, e.g., due to operations in remote locations. Thus, once deployed, the UAVs cannot consistently rely on any online third parties.

Moreover, we assume that the UAVs integrate an instance of our proximity detection protocol, namely LPPD. Therefore, the UAVs can exchange wireless messages to detect co-location and take appropriate maneuvers to avoid collisions. To this aim, each UAV u_i stores a private/public key pair used for protocol execution. It is reasonable to assume that this key pair has been released and installed by the manufacturer, together with a public key certificate C_i. Note that the certificate is installed at manufacturing time and does not force any third party to be available online during the execution of the protocol. Although we acknowledge that maintaining a unique Public Key Infrastructure (PKI) worldwide might be impractical—though doable, especially if provided by international aviation-related bodies, like IATA—, that is certainly possible on a regional level, which is where our scenario mainly applies. Finally, we report the used notations in Tab. 7 (Annex).

2.2 Adversary and Threat Model

The adversary assumed in our work, namely $\mathcal{A}$, features both passive and active features. On the one hand, $\mathcal{A}$ is a global eavesdropper, equipped with multiple antennas to detect and decode packets transmitted on the same channel(s) used by the legitimate UAVs, i.e., $B = [2.4 - 2.5]$ GHz. We also assume that $\mathcal{A}$ can deploy the receiving antennas in a vast geographical area, possibly synchronize such receivers, and then process the receiving packets locally. In this sense, we also assume that $\mathcal{A}$ has no particular constraint in its computational capabilities. On the other hand, we assume that $\mathcal{A}$ also features active capabilities. Thus, it can replay the recorded packets or inject its own (forged) packets, possibly falsifying some ad-hoc fields in the messages (i.e., the sender identity), to avoid detection by intrusion detection systems.

Moreover, we assume that $\mathcal{A}$ does not have any prior knowledge of the location of the UAV at the time of the attack. Neither it can *visualize* the UAVs a priori, before the execution of the protocols discussed below—if one of the two above-cited conditions apply, there would be no possible defense. At the same time, note that the above-referred conditions do not restrict the application of our solution for proximity detection. Indeed, Wi-Fi radio chipsets adopted on UAVs feature reception ranges up to 7 km. As reported in [34] and [20], attacks against Wi-Fi systems can be conducted even outside their nominal reception range, making our solution valuable for proximity detection even when the two communicating entities are not in mutual visibility.

We also assume that $\mathcal{A}$ could consist of multiple colluding entities, possibly geographically distributed over the reception area. For example, $\mathcal{A}$ could adopt RSS or time-based wireless localization techniques to estimate the location of a particular transmitter by resorting to an entirely passive receiver infrastructure [2, 23]. In this context, we highlight that $\mathcal{A}$ does not have any prior information about the number of UAVs in its reception area. When coupled with the minimal amount of wireless messages transmitted by LPPD, the previous feature nullifies any attempts by $\mathcal{A}$ to localize the emitting source through dedicated time-based or power-based localization techniques.

The threats enabled by $\mathcal{A}$ to the UAV can be manifold, e.g.: (i) disrupt the flight of the UAV, e.g., via jamming or spoofing; (ii) capture the UAV, e.g., via a net; and, (iii) inferring on the travelled trajectory, e.g., to deduce if it has visited (or not) a specific location. Note that for (i) and (ii), the adversary implicitly needs to know the UAV position, while for (iii), such a need is a truism. Our solution helps keep such data private, thwarting the cited adversary's objectives.

3 THE LPPD PROTOCOL

In this section, we first introduce our solution in a nutshell (Sec. 3.1). Then, we explain the logic of the space tessellation (Sec. 3.2) and, finally, we describe the full LPPD scheme (Sec. 3.3).

3.1 Our Solution in a Nutshell

Our solution, namely LPPD, allows two directly connected UAVs, in mutual radio visibility, to know if they are *in proximity*, i.e., at risk of immediate physical collision, without revealing their actual locations. To this aim, LPPD builds on two enabling components. The first component is a random one-time space tessellation generated by the client UAV (i.e., the UAV initiating the protocol), which allows us to uniquely identify the geographical area where the UAV is and its neighborhood, without revealing the location of the UAV. We provide the details of the logic of the space tessellation used in LPPD in Sec. 3.2. The second enabling component of LPPD is a private-set intersection protocol inspired by the solution in [9]. Such a protocol allows one entity (the client device) to know which elements of the set in its possession match the ones possessed by the remote entity (the server device) without revealing anything else to both entities. Note that the straightforward application of the primitive proposed in [9] to our scenario and problem (proximity between aerial vehicles) would not be a satisfactory solution. Indeed, location data are continuous and affected by an error due to the inaccuracy of the measurement technology (e.g., the GPS technology), making our problem interesting and challenging at the same time. The entire protocol integrating the space tessellation logic with the private-set intersection algorithm is described in Sec. 3.3. The UAVs could report an alarm to the pilot if proximity is detected. Then, the remote pilot can reduce speed and adopt any evasive maneuvers—as proximity detection is our objective, how to achieve collision avoidance is out of scope. Finally, note that in case of proximity, the location privacy of the neighbor UAV would be broken: such an outcome is deemed acceptable since safety is prioritized over privacy.

3.2 Rationale of the Space Tessellation

LPPD is rooted in a specific division of the Earth's surface in multiple dynamic three-dimensional spheres, whose logic is explained below.

Let us consider the UAV u_α, located at $\mathsf{pos}_\alpha = (x_\alpha, y_\alpha, z_\alpha)$. Given that u_α is equipped with a GNSS receiver (e.g., GPS), its estimated location will likely be affected by an error, depending on the specific location and satellites' visibility. We assume that δ is the maximum error on the location of u_α caused by GNSS inaccuracy. When running LPPD, u_α would like to ensure that the remote entity u_β is not located closer than a threshold distance T_α from its current position. Naming $d_{\alpha,j}$ the distance between u_α and u_j, they are not at risk of immediate collision if Eq. 1 holds.

$$d_{\alpha,j} \geq \delta + T_\alpha. \tag{1}$$

The model in Eq. 1 can be extended to consider other drone movements, such as its acceleration/deceleration. However, we do not explicitly address these extensions to keep our discussion generally applicable.

Thus, it is possible to draw a sphere, centered at the location of u_α, namely pos_α, with radius $r_\alpha = \delta + T_\alpha$. Then, Eq. 1 is satisfied only if the location of u_j does not fall into such a sphere. Eq. 1 could be further refined, considering the expected protocol execution time, namely t_p [s]. In particular, considering the maximum possible speed of a UAV, namely V_{MAX} [m/s], during protocol execution, the remote entity might move for a maximum distance of $V_{MAX} \cdot t_p$. Thus, Eq. 1 can be extended to obtain Eq. 2.

$$d_{\alpha,j} \geq \delta + T_\alpha + V_{MAX} \cdot t_p. \tag{2}$$

Therefore, considering Eq. 2, $r_\alpha = \delta + T_\alpha + V_{MAX} \cdot t_p$. We denote r_α as the *guard radius*, as it identifies the minimum allowed

displacement between u_α and any generic remote entity u_j. Thus, at each protocol run, u_α maps its position as the center of a sphere having radius r_α. Then, u_α needs to generate a (random) identifier of its location to be used for proximity detection. To this aim, it extracts three nonces $\mathsf{s} = s_x, s_y, s_z$, with $s_x, s_y, s_z \in Z_n$ and n sufficiently large ($n \geq 64$ bits), and uses them to obtain a random origin point of the space $\mathsf{O}_\alpha = O_x, O_y, O_z$ (Eq. 3), and to translate its location (Eq. 4).

$$\begin{cases} O_x = x_\alpha - s_x \cdot d_{\alpha,j}, \\ O_y = y_\alpha - s_y \cdot d_{\alpha,j}, \\ O_z = z_\alpha - s_z \cdot d_{\alpha,j}. \end{cases} \tag{3}$$

$$\begin{cases} x'_\alpha = x_\alpha - O_x, \\ y'_\alpha = x_\alpha - O_y, \\ z'_\alpha = z_\alpha - O_z. \end{cases} \tag{4}$$

The location identifiers $\tilde{x_\alpha}, \tilde{y_\alpha}, \tilde{z_\alpha}$ are then obtained as per Eq. 5.

$$\begin{cases} \tilde{x_\alpha} = \left\lfloor \frac{x'_\alpha}{2 \cdot d_{\alpha,j}} \right\rfloor, \\ \tilde{y_\alpha} = \left\lfloor \frac{y'_\alpha}{2 \cdot d_{\alpha,j}} \right\rfloor, \\ \tilde{z_\alpha} = \frac{z'_\alpha}{2 \cdot d_{\alpha,j}}. \end{cases} \tag{5}$$

We will denote $pos_\alpha = \mathcal{H}(\tilde{x}_\alpha, \tilde{y}_\alpha, \tilde{z}_\alpha)$ the unique location identifier of u_α for the current instance of the protocol, being $\mathcal{H}$ an hashing function.

We remark that the nonces randomize the location of u_α, which otherwise would always be $[0, 0, 0]$. Thus, with the computations in Eq. 3, Eq. 4, and Eq. 5, the actual location of u_α is still at the center of a sphere, but the specific identifier of the sphere is moved according to the nonces s. Finally, note that u_α should deliver $\mathsf{O}_\alpha = O_x, O_y, O_z$, to allow the remote entity to map its location in the related location identifiers (only the points are delivered, while nonces are kept secret). Then, the comparison among the identifiers occurs in the encrypted domain, using a private-set intersection scheme based on RSA (see below).

Note that, due to the specific shape of the WGS84 ellipsoid, some locations (e.g., at the poles) might appear very different from circles. Such imperfections are not an issue for our protocol, as they can be approximated through the smallest circles where such shapes can be inscribed.

3.3 Detecting proximity using LPPD

Assume a UAV u_α is flying in a given area, and it becomes aware of the presence of another UAV, namely u_β, located in its reception range. Such a discovery can be achieved in several ways, e.g., by detecting the presence of the video stream of the UAV, the communication between the UAV and its controller, or periodical broadcast messages emitted by the UAV. After detection, u_α and u_β establish a secure connection, used to exchange public parameters, i.e., the public keys (n_α, e_α), (n_β, e_β) and, if they do not trust each other, the public key certificates C_α and C_β. Then, they run LPPD over such a secure connection to verify they are not at risk of immediate collision. To this aim, they execute the operations reported in Fig. 2, as explained below.

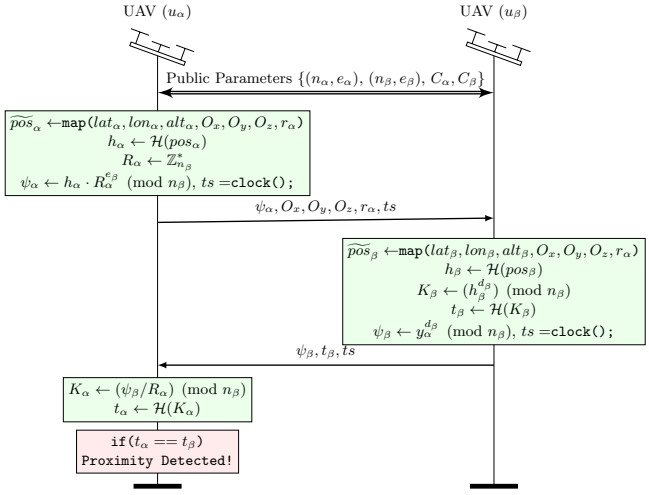

Figure 2: Sequence Diagram of LPPD.

Note that the UAVs do not compute each other's trajectories but only check if their locations fall within the same sphere, revealing only the matching sphere (if any).

- To start LPPD, u_α selects the space tessellation parameters, as described in Sec. 3.2. Thus, it generates the *origin point* O_α, and the guard radius r_α, used to identify its *safe area* and to allow remote entities to map their position into its space tessellation, respectively. Then, it maps its own location $\mathsf{pos}_\alpha = (lat_\alpha, lon_\alpha, alt_\alpha)$ to a bit-string value pos_α, i.e., the spherical tile identifying its position.
  Then, u_α executes the following: (i) computes $h_\alpha = \mathcal{H}(pos_\alpha)$, i.e., the hash of the tile identifier pos_α; (ii) extracts a random value $\rho_\alpha \in \mathsf{Z}^*_{n_\beta}$, where n_β is the RSA modulus contained in the public key of u_β; and, (iii) generates an encrypted location identifier, namely, ψ_α, as per Eq. 6.

$$\psi_\alpha = h_\alpha \cdot \rho_\alpha^{e_\beta} \pmod{n_\beta}. \tag{6}$$

  Finally, u_α delivers wirelessly the following elements:
  - the encrypted location identifier ψ_α;
  - the one-time *origin point* of the space tessellation O_α;
  - the guard radius r_α;
  - the expiration time of the message, namely, ts_α.
- When u_β receives the message from u_α, it executes the following operations. First, u_β maps its own location $\mathsf{pos}_\beta = (lat_\beta, lon_\beta, alt_\beta)$ into the mapping space instructed by α, uniquely identified by the origin point O_α and the guard radius r_α. The mapping results in a location identifier pos_β, i.e., the identifier of the current location of the UAV u_β into the tessellation space instructed by u_α. Then, u_β computes $h_\beta = \mathcal{H}(pos_\beta)$, i.e., the hash of the location pos_β, and it generates the parameter K_β, as per Eq. 7.

$$K_\beta = h_\beta^{d_\beta} \pmod{n_\beta}. \tag{7}$$

Third, u_β computes the tag digest t_β, as $t_\beta = \mathcal{H}(K_\beta)$, and it generates the encrypted location identifier ψ_β as per Eq. 8.

$$\psi_\beta = \psi_\alpha^{d_\beta} \pmod{n_\beta}, \tag{8}$$

where d_β is its private key. Finally, u_β sends:
- the encrypted location identifier ψ_β;
- the tag digest t_β;
- the expiration time of the message ts_β.

- At reception, u_α executes the following. First, it executes the modular division reported in Eq. 9.

$$K_\alpha = (\psi_\beta / \rho_\alpha) \pmod{n_\beta}, \tag{9}$$

Then, u_α computes the tag digest $t_\alpha = \mathcal{H}(K_\alpha)$. Note that, in case $t_\alpha = t_\beta$, i.e., the UAVs u_α and u_β mapped their locations into the same spherical tile, then $h_\alpha = h_\beta$, and then, Eq. 10 holds (operations are $\pmod{n_\beta}$).

$$K_\alpha = (\psi_\beta / \rho_\alpha) = (h_\alpha \cdot \rho_\alpha^{e_\beta})^{d_\beta} / \rho_\alpha = h_\alpha^{d_\beta} = K_\beta \tag{10}$$

Thus, when $K_\alpha = K_\beta$, u_α realizes to be in proximity of u_β. Otherwise, when $t_\alpha < t_\beta$, then $K_\alpha < K_\beta$, meaning that the UAVs are not in proximity.

Therefore, at the end of LPPD, the initiator u_α knows to be in proximity of u_β, and it needs to take action to avoid a collision. The initiator also has to inform the responder of the detected proximity. Such activity can be done through a dedicated message delivered over a regular Wi-Fi connection, secured via TLS. No privacy measures apply here, as safety takes priority over location privacy. The following actions may be heterogeneous. For instance, u_α can contact u_β, and they both can agree on a shared path, not leading to physical collisions. Such an interaction will likely imply an agreement over a set of explicit location identifiers. Indeed, avoiding collisions when the entities are in proximity takes priority compared to possible (reduced) privacy leakage due to the high chance of severe immediate threat to safety. Alternatively, u_α can autonomously take an evasion maneuver, minimizing the risk of collisions, without interacting further with u_β. Without loss of generality, the actions to be taken after discovering the co-location depend on the specific nature and features of the UAV (e.g., level of autonomy, presence, and type of controller).

4 SECURITY ANALYSIS

In this section, we discuss the security features offered by LPPD. (Sec. 4.1) and provide the formal verification of such properties through *ProVerif* (Sec. 4.2).

4.1 Security Services

Location Privacy. The most critical security service offered by LPPD is its capability to guarantee the detection of proximity between two UAVs in wireless visibility without revealing their respective locations. The cited property is achieved through a randomized space tessellation and an algorithm for private-set intersection. Although the cardinality of the possible location identifiers is limited and possibly subject to brute-force attacks, we notice that brute-forcing the entire space of the location identifiers is not possible. Indeed, the encrypted location identifier delivered by u_β is uniquely tied to its private key d_β, n_β , the encrypted location identifier sent by u_α and, in turn, to the nonces $\mathtt{s}$ selected by both the entities. Provided that the nonces have sufficiently large entropy, the search space for an adversary would be large enough to be considered unfeasible. Moreover, even assuming to know the nonce used to randomize location identifiers, an entity would need to have all the possible encrypted location identifiers for one entity, which is unfeasible for an interactive protocol and easily detectable by the device subject to such an attack. Also, note that more than the straightforward application of a private-set intersection protocol would be required to provide location privacy in our context. Indeed, without randomizing the space tessellation, an honest-but-curious adversary would always know that the location of the entity initiating the protocol (u_α) is coincidental with the origin point of the space tessellation. Instead, LPPD randomizes the origin point of the space tessellation at each run with dedicated private nonces, preventing any guessing by remote parties. Therefore, under standard security assumptions, the UAVs' locations are not revealed while still allowing for proximity detection. We formally prove this property in Sec. 4.2 through *ProVerif*.

Protection against Active Attacks. An active attack might try to interfere with the regular execution of LPPD by injecting malicious messages. Such messages can be replayed after being eavesdropped or forged ad-hoc to impersonate the legitimate parties. First, we notice that LPPD runs after establishing a secure connection. Thus, it relies on the previously-run secure session establishment protocol to derive a session key known only to the legitimate parties. The usage of such a key allows for authenticating the parties, as well as detecting and rejecting impersonation attacks. Moreover, fresh nonces and timestamps are generated ex-novo for each new instance of the protocol. Thus, the validity period included in the LPPD messages allows automatically rejecting messages replayed after expiration. Overall, beyond protecting the location privacy of the involved parties, the scope of such values is also to guarantee immediate identification of replays to detect and reject them. We formally prove this property in Sec. 4.2 through *ProVerif*.

Wireless Localization Attacks. An adversary might try to localize the emitting source using wireless localization techniques. A passive adversary can deploy an array of antennas over a large area and analyze physical-layer features such as the Time of Arrival (ToA) or the Received Signal Strength (RSS) of the LPPD messages jointly to obtain a location estimate. Note that such solutions usually require a significant number of packets to estimate the transmitter's location, and such a number drastically increases with the mobility of the emitting source and the level of noise affecting the communication link. As shown through experiments in Sec. 5.3, LPPD requires a minimal number of wireless messages (as little as 1 per device in the most lightweight configuration), thwarting any effort to localize the involved entities based on physical-layer features. When more messages are needed, such as when higher security is desired, several solutions for thwarting localization are available. For instance, the UAs might randomly change the transmission power of LPPD messages and use (pseudo)-random pseudonyms, known only to the remote entities involved in the protocol, to make it more difficult and expensive for a remote passive attacker to identify packets coming from the same source and apply localization attacks.

4.2 Formal Verification

We verified the most important security objectives of LPPD, i.e., the secrecy of the location of the involved entities and the mutual authentication among involved peers, through the automated verification tool *ProVerif* [7], in line with many recent scientific contributions on network security [17] [32]. Note that the security of the atomic building blocks of LPPD has already been verified formally in the past, e.g., by the authors in [9]. Thus, a rigorous formal analysis of their security would not add anything new to the current state of the art. Conversely, the composition of such blocks into a new protocol might create novel vulnerabilities. Thus, proving the security of LPPD is a must. In accordance with a vast corpus of the literature, we choose to prove its security via symbolic methods, selecting *ProVerif* as a tool.

The formal verification tool *ProVerif* assumes the Dolev-Yao attacker model, i.e., it allows the attacker to read, modify, delete, and forge packets and inject them into the public communication channel. Under the cited assumptions, *ProVerif* can verify if the designed protocol achieves the claimed security goals defined by the user. Where an attack is found, *ProVerif* also describes the attack steps. We verified the security of LPPD in *ProVerif* assuming two communicating UAVs u_α and u_β, as discussed in Sec. 3. Specifically, we tested three security features: (i) the secrecy of the location identifiers of u_α and u_β during the protocol execution; (ii) the resistance of the protocol to offline guessing attacks on the locations pos_α and pos_β, originated by the inherent low entropy of such information; and, (iii) the authenticity of the messages of the UAVs. Therefore, according to the logic of the *ProVerif* tool, we defined four main events.

(1) *acceptUAVa(x,y)*: Indicating that UAV u_β believes it has initialized a protocol instance with UAV with ID $y = u_\alpha$ and the supplied symmetric key x.
(2) *acceptUAVb(x,y)*: Denoting that UAV u_α believes it has initialized a protocol instance with UAV with ID $y = u_\beta$ and the supplied symmetric key x.
(3) *termUAVa(x,y)*: Indicating that UAV with ID $y = u_\beta$ believes it has finalized the protocol with UAV u_α and the supplied symmetric key x.
(4) *termUAVb(x,y)*: Denoting that UAV with ID $y = u_\alpha$ believes it has finalized the protocol with UAV u_β and the supplied symmetric key x.

In line with the logic of *ProVerif*, we verified that message authenticity holds through verifying security properties such as *mutual authentication* and *impersonation resistance*. Recalling that the UAV u_α shares its secret key with the UAV u_β, it follows that, if the UAV u_α completes the protocol, this latter one believes to have done so with the UAV u_β, and hence authentication of the UAV u_β to the UAV u_α holds. Note that the UAV u_β completes the protocol successfully only when the UAV u_α initiates it. Indeed, if UAV u_α believes it has terminated the protocol with the UAV u_β, this latter one is the correct entity executing the protocol, and vice-versa. Thus, the UAVs u_α and u_β are mutually authenticated. In addition, we recall that, for the security properties we want to verify, *ProVerif* provides the output *not attacker(val[]) is true* when the attacker cannot derive the value of *val*. In contrast, it provides the output *not attacker(val[]) is false* if the attacker can do so. Similarly, *ProVerif* provides the output *weak secret(val[]) is true* when the attacker cannot distinguish a correct guess of the value *val* from an incorrect guess, while it provides the output *weak secret(val[]) is false* when offline guessing attacks on *val* are possible.

```
Verification summary:
Weak secret posA is true.
Weak secret posB is true.
Query    inj-event(termUAVa(x, y))        ==>
inj-event(acceptUAVb(x, y)) is true.
Query    inj-event(termUAVb(x, y))        ==>
inj-event(acceptUAVa(x, y)) is true.
Query not attacker(posA[]) is true.
Query not attacker(posB[]) is true.
```

Figure 3: Excerpt of the output provided by the *ProVerif* tool.

Fig. 3 shows the excerpt of the output of the *ProVerif* tool when executed on a local machine. The output of the tests shows that: (i) the way pos_α and pos_β are used in LPPD protects them against offline guessing attacks, originating from their low entropy; (ii) the locations of the entities, i.e., pos_α and pos_β, are not exposed to the attacker; and, (iii) message authenticity always holds. Note that we also mitigated possible replay attacks by verifying the freshness of the timestamp included in the messages.

Thus, LPPD can effectively protect the locations of the UAVs, ensuring effective proximity detection properties and enabling collision avoidance. We also released the source code of the implementation of LPPD in *ProVerif* as open-source [31], to allow interested readers to verify our claims and further re-use our code.

5 PERFORMANCE EVALUATION

In this section, we first evaluate the effectiveness of LPPD (Sec. 5.1), then we provide its implementation details (Sec. 5.2), and finally, we report the results of real experiments (Sec. 5.3).

5.1 Simulation Analysis

The most critical objective of LPPD is to detect proximity among UAVs. This task is achieved thanks to the space tessellation logic introduced in Sec. 3.2. To test the effectiveness of LPPD in detecting all the possible co-locations, we set up a simulation analysis using Matlab R2021a. Specifically, we took a worst-case condition by configuring 50 UAVs to move randomly in a geographical area of $50 \times 50 \times 120\ m^3$, at a random speed modulus in the interval $[0 - 20.88]$ meters per second (coincidental with 50 km/h). In line with the GPS accuracy in open space, we assumed that the UAV locations are affected by a random error, uniformly extracted in the interval $[0 - 2]$ meters for the horizontal components and $[0 - 3]$ meters for the vertical component, to have $\delta = 0.375$ meters. As mentioned in Sec. 2.1, when the drone is in a location where a more significant error is expected, the value of δ can be fine-tuned to guarantee zero false negatives, following a safety-first approach. We also defined a common guard space of 5 meters for all the UAVs. Note that the above-specified parameters are just exemplary, and LPPD can work effectively with any of their values.

With such parameters, the *guard radius* amounts to $\delta+T+V_{MAX} \cdot t_p = 0.375 + 5 + 20.88 \cdot 0.02 = 5.793$ meters. In such a scenario, we evaluated the capability of the space tessellation logic of LPPD to identify approaching co-locations while not leading to the invasion of the related *guard radius*. To this aim, we define the *Proximity Detection Ratio* as the ratio between the number of detected co-locations and the number of co-locations that actually happened and led to an invasion of the guard radius (ground truth). Fig. 4 shows the proximity detection ratio as a function of the sphere's radius used for space tessellation. Each value in the figure is an average of 10,000 seeds (we also report the 95% confidence intervals). As intuition suggests, increasing the sphere radius increases the

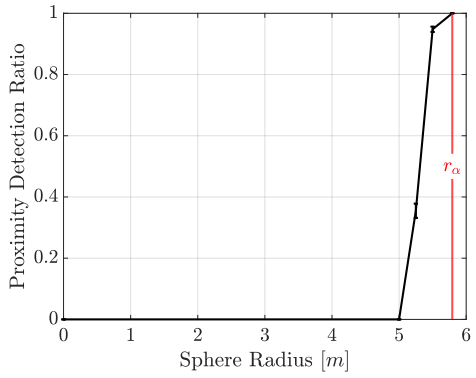

Figure 4: Proximity Detection Ratio of LPPD, varying the radius of the sphere r_α used for space tessellation.

capability of LPPDto detect co-locations before such events lead to the invasion of the guard radius. Values lower than the guard radius lead to zero co-locations detected while increasing the value of the guard radius over the safety guard enhances the performance. However, values very close to Eq. 2, such as $r_\alpha = 5.5$ meters, report imperfect proximity detection ratios (0.947), as they cannot handle drone movements at the maximum possible speed. We can obtain the value 1 for the proximity detection ratio only setting r_α as per Eq. 2 (or higher), guaranteeing total safety. Note that, with such a choice of r_α, some UAVs might be identified as co-located although being quite far (very close to r_α m)—we might consider this case as a false positive. However, as the priority of LPPD is safety, it always identifies potential co-location at the required resolution (no false negatives), leaving the pilots to assess the warning. Finally, protocol scalability in very dense deployments might be an issue. We chose the area and number of drones described above for a fair comparison with [30], which uses the same parameters. In very dense deployments, drones will likely be too close to each other and give up privacy to prioritize safety, i.e., not use LPPD.

5.2 Implementation Details

We implemented a prototype of LPPD on the 3DR-Solo hardware [5]. The 3DR-Solo drone includes a CPU *i.MX6 Solo* designed by *Freescale System*, compatible with the de-facto standard *Pixhawk* autopilot. It is powered by a single-core processor ARM Cortex A9, running at 1.00 GHz, and it is equipped with 7,948 MB of ROM and 512 MB of RAM. Moreover, this drone features modules for Cryptographic Acceleration and Assurance (CAAM) and True and Pseudo-Random Number Generator (certified by NIST), which are useful for executing cryptography primitives efficiently. As the operating system, the 3DR-Solo runs 3DR Poky Linux, based on the Linux project Yocto [3]. We implemented LPPD in *C*, within the stock *3DR Poky* OS, version 1.5.1. In particular, we integrated LPPD within the Micro Air Vehicle Link (MAVLink) 1.0 communication protocol [1], using the lightweight Micro Air Vehicle Message Marshalling Library [4]. We recall that MAVLink is a lightweight UDP-based messaging protocol adopted by several large and small unmanned vehicles, enabling communication between drones, their onboard components, and ground control stations. We also recall that MAVLink allows full customization of messages, from a minimum packet length of 8 bytes (for acknowledgements) up to a maximum of 263 bytes.

We report the structure and content of the MAVLink frame (customized for LPPD) in Tab. 1. We used the standard format *float32* for encoding GPS coordinates. Converting the GPS coordinate to a *float32* might generate small inaccuracies. On the one hand, internal calculations might use a more precise format, e.g., *float64*. On the other hand, when this is not possible, such implementation inaccuracy can be included in the value of δ, making our model still valid and applicable for proximity discovery. To be compliant with the MAVLink standard, we extended MAVLink with a dedicated Message ID (`0xCA`). We used the RSA algorithm for encryption and decryption operations on big (modular) integers by integrating the functions provided by the popular OpenSSL library, ver. 1.0.0 [21].

For the experimental evaluation, we selected four reference RSA key sizes, i.e., *RSA 1024, RSA 2048, RSA 3072* and *RSA 4096*, providing security levels equivalent to 80, 112, 128 and 140 symmetric key bits, respectively [6], to provide an adequate level of security even for critical scenarios. Moreover, we adopted *SHA-1* as the hashing function and cryptographic Pseudo Random Number Generator (PRNG) (*/dev/urandom*) seeded with 2,048 bits. Our implementation on the 3DR-Solo requires 1,545.324 KB of Flash Memory (with a static linking of the libraries) and 90.179 KB of RAM. Finally, we remark that our implementation leverages popular open-source tools, such as the Poky OS, MAVLink, and OpenSSL, supported by a large variety of commercial UAVs.

5.3 Performance Assessment

We performed many experiments using the implementation discussed in Sec. 5.2, aimed at measuring the cost of integrating and running LPPD on a real UAV, in terms of processing time, bandwidth, and energy consumption.

As a first important investigation, Fig. 5 reports the average duration of the basic modular operations required by LPPD, when executed on the 3DR-Solo drone, over different key sizes. The figure also reports the 95% confidence interval, computed over 1,000 tests. Specifically, taking as a reference the modulus size 1024, we notice that the modular multiplication operation can be completed on average in only 0.02433 ms. The modular divisions are a little more expensive, being executed in 1.177 ms, on average. Finally, modular exponentiations are completed in an average time of 16.82 ms, being the most expensive ones. Consider that the key size of 1024

Table 1: MAVLink Frame and LPPD Payload Notation.

Field	Type	Size [B]	Description
SOF	`uint8_t`	1	MAVLink 1.0. Start of Frame, set to `0xFE`.
LEN	`uint8_t`	1	Payload Length, in bytes.
SEQ	`uint8_t`	1	Sequence number.
SID	`uint8_t`	1	UAV System Id. number.
CID	`uint8_t`	1	System Id. number of the transmitting component.
MID	`uint8_t`	1	Message Type Id. number, set to `0XCA`.
Payload	`*uint8_t`	`0-255`	LPPD message.
CRC	`uint8_t[2]`	2	Checksum.
LPPD Payload			
KIX	`uint8_t`	1	Key Index.
RAD	`uint8_t[2]`	2	Sphere Radius r_i, in centimeters.
O	`float32[3]`	12	WGS84 origin coordinates (4 bytes each), in degrees.
ENC	`*uint8_t`	`128-532`	Encrypted data (size based on RSA keys and hash).
TS	`uint32_t`	4	Message Timestamp, in UTC format.

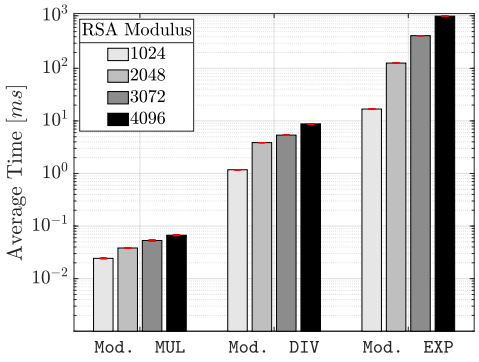

Figure 5: Time to execute modular operations on the 3DR-Solo, considering different RSA key lengths. Red bars report 95% confidence intervals.

bits provides an equivalent security level of 80 bits, recommended by the US-based NIST for regular security operations. Thus, the above-discussed results demonstrate that modular atomic operations required by LPPD can be executed almost in real-time on a modern commercial UAV, not introducing large delays that would be hardly manageable in the execution of our scheme. When necessary, e.g., for very-sensitive operations, larger key sizes can also be used, providing enhanced security at the cost of a noticeable increase in the computational and bandwidth overhead of the scheme, due to the increasing size of the cryptography elements to be exchanged between the involved entities.

With specific reference to the proposed LPPD scheme, we first measured the time needed to run a single instance of LPPD on the 3DR-Solo, by considering the previously-mentioned key sizes. We report in Fig. 6 the average time required to execute LPPD over 1,000 tests (with 95% confidence intervals). In the figure, we explicitly considered the separate contributions of the processing (packet generation, cryptography operations) and radio operations. Note that the average computation time reported in the graph below includes also the time to acquire the GPS location, in a scenario where the current almanac, initial position, Ephemeris and time information of the GPS are all set. Considering the minimum re-

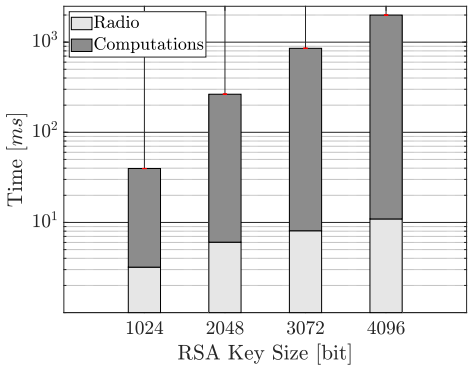

Figure 6: Time to execute LPPD on the 3DR-Solo, with different key sizes.

quired security level of 80 bits, corresponding to an RSA key size of 1,024 bits, LPPD completes on the client in only $\approx$ 39.66 ms, confirming to be a viable option for efficient proximity detection. In line with the previous results, the execution of LPPD with increased key size leads to increased completion times. As specified in Sec. 3.2, the above-considered completion times should also be taken into account by an UAV when establishing its *guard radius*. With the modification in the previously-mentioned Eq. 2, the client UAV can be sure that, even assuming the worst-case speed of the remote entity, the approaching UAV cannot come closer than δ (meters) from its actual location at the end of the execution of the protocol, further boosting the effectiveness of LPPD in detecting proximity. Clearly, with such a modification, LPPD could also lead to more false positives, i.e., detection of proximity even when the distance between the vehicles is less than r_α but not less than δ. We believe that the probability of false positives is acceptable in UAV-related use cases, where preventing collisions and safeguarding safety has the highest priority.

We also evaluated the bandwidth overhead incurred by the integration of LPPD. Specifically, Fig. 7 reports the total number of MAVLink messages required by LPPD, based on the adopted key size. Additional configuration details are provided in Tab. 2. With the most lightweight configuration (key size of 1,024 bits), LPPD requires only 2 messages, i.e., 1 frame per UAV. Due to the increasing size of the cryptography materials, more messages are needed with higher security levels, leading to additional energy consumption (see below).

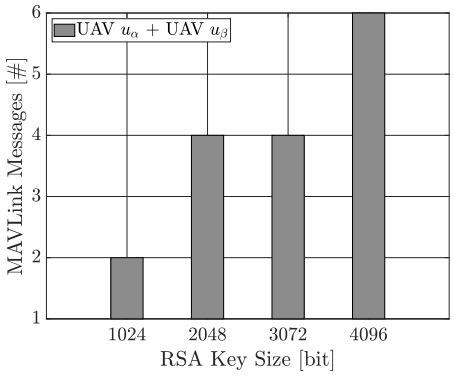

Figure 7: Messages required by LPPD between two UAVs.

Table 2: Payload size and no. of messages required by LPPD, with different RSA key sizes.

Key Size (bits)	Message size u_α [B]	Message size u_β [B]	Total Messages
1024	146	153	2
2048	274	281	4
3072	402	409	4
4096	530	537	6

To measure the energy consumption demanded by running a single instance of LPPD, we identified the contributions to the consumption exacted by computations and radio operations. To obtain the energy consumption related to the CPU operations, we used the telemetry data conveyed by the 3DR-Solo to the remote controller through the MAVLink protocol. In detail, we measured the difference in the electrical current drained by the drone between two different states: (i) at rest; and, (ii) during the execution of LPPD. As a result, we obtained an average difference of $\approx$ 20 mA in the electric current drained by the drone, over 1,000 runs.

To estimate the energy consumption of the radio operations, we considered that the radio chip on-board of the 3DR-Solo drone is a chip of the family *AR9300*, working with an input voltage of 3.3 V, consuming 296.970 mA in TX mode and 187.879 mA in RX mode with the IEEE 802.11b protocol [16]. We also assumed that a generic packet transmitted by our drone is modulated through the standard Direct Sequence Spread Spectrum (DSSS) technique, using the Differential Binary Phase-Shift Keying (DBPSK) modulation, a transmission rate of 1.0 Mbps on a 22 MHz channel bandwidth, and a short guard interval of 800 ns. Then, we combined the contributions of the processing and radio chip to obtain the overall energy consumption of LPPD, through Eq. 11.

$$E[mJ] = V \cdot \int_0^T i(t)dt, \tag{11}$$

where V is the input voltage (15.11 V for the UAV's battery and 3.3 V for the radio chip) and $i(t)$ the instantaneous drained current (additional 20 mA required by LPPD on the UAV's battery and 296.970 mA for the radio chip). Fig. 8 reports the average results of our experiments, together with the 95% confidence interval, computed over 1,000 tests. We also summarize in Tab. 3 the time and energy consumption reported in Fig. 6 and Fig. 8.

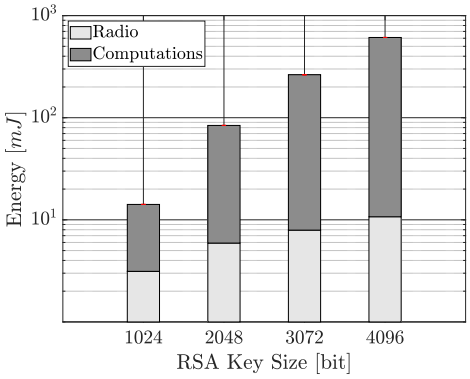

Figure 8: Energy for executing LPPD on the 3DR-Solo.

Table 3: Avg. time and energy (with 95% confidence intervals) required to execute LPPD, with different RSA key size.

Key size (bits)	Radio Time (ms)	Comp. Time (ms)	Radio Energy (mJ)	Comp. Energy (mJ)
1024	3.192	36.471 ± 0.045	3.128	11.022 ± 0.01
2048	6.040	258.481 ± 0.104	5.919	78.113 ± 0.03
3072	8.088	846.6 ± 0.171	7.926	255.843 ± 0.05
4096	10.936	1,985.642 ± 0.297	10.717	600.061 ± 0.09

Considering the most lightweight configuration (modular size of 1,024 bits), LPPD consumes only $\approx$ 14.15 mJ per instance, confirming its little impact on the regular operations of a UAV. Indeed, given that the overall capacity of the battery powering the 3DR-Solo drone is 282,860 J (5,200 mAh), assuming to work with a key size of 1,024 bits, LPPD consumes on average only $5 \cdot 10^{-6}$% of the battery of the drone for each instance, further demonstrating its limited overhead and thus emerging once more as a lightweight and efficient solution for proximity detection. Moreover, in order to save energy, the pilot of the drone can still decide to disable LPPD in some situations, e.g., when the remaining energy falls below a threshold or when flying in areas with a very rare presence of drones, trading off energy with privacy.

6 RELATED WORK AND COMPARISON

In the last years, several works investigated proximity detection and collision avoidance on UAVs.

Typically, researchers provided solutions for the above-cited issue in the context of a swarm of drones, moving together to

Table 4: Qualitative comparison of LPPD against state-of-the-art approaches for proximity discovery on UAVs.

Ref.	No ad-hoc sensors	No calibration	Support for high-speed UAVs	Support for unknown UAVs	No ad-hoc ground infrastructure	Location Privacy Support	Support for RPASs
DJI [10]	✓	✓	✓	✓	✓	–	✓
Wang et al. [36]	–	–	–	✓	✓	–	–
Beard et al. [22]	✓	✓	✓	–	✓	–	–
Shen et al. [27]	✓	✓	✓	–	✓	–	–
Sabetghadam et al. [24]	✓	✓	–	–	✓	–	–
Park et al. [15]	–	–	–	✓	✓	–	–
Gageik et al. [19]	–	–	–	✓	✓	✓	–
Watanabe et al. [37]	–	–	–	✓	✓	✓	✓
Hsu et al. [13]	✓	✓	–	✓	–	✓	–
PPCA [30]	✓	✓	✓	✓	✓	✓	–
PPTM [25]	✓	✓	✓	✓	✓	✓	–
LPPD	✓	✓	✓	✓	✓	✓	✓

accomplish a mission. A few examples in this direction include the works by the authors in [27], [24], [22], [36], and [15]. The first ones to investigate on the issue were the authors in [22], who developed collision avoidance algorithms for the team and individual drone optimality, without taking into account privacy considerations. Similar considerations apply to the work by the authors in [36], adopting control-theory based strategies to timely detect and avoid collisions of drones in swarms with obstacles. Similarly, the solution proposed in [15],[27], and [24] addressed the problem of collision detection and avoidance collaboratively, by letting the involved entities share information about the travel path in the swarm, not taking privacy considerations into account.

To address collision avoidance, many approaches such as [19] and [37] use on-board complementary sensors to detect approaching collision with any object. On the one hand, such approaches do not consider the knowledge of UAVs locations, thus preserving location privacy. On the other hand, they often require calibration before deployment for each new mission, and they are ineffective when the potential speed of the approaching vehicle is high.

Recently, the authors in [13] implemented a technique based on reinforcement learning to detect and avoid collisions on UAVs. The scheme leverages the interaction of the UAV with a ground IoT control network and forces the UAV to run across the IoT devices in a specific way. Thus, forcing the deployment of a collaborative IoT control network, where an operator controls the scenario, the solution is impractical in mission-critical scenarios.

It is also worth mentioning the solution recently introduced by the commercial drone manufacturer DJI, allowing drones to broadcast via Wi-Fi information such as the location, altitude, speed, direction, and identification number of the drone directly to smartphones, so as to detect collisions in advance [10]. Clearly, such a solution is not privacy-preserving.

We also highlight that our solution integrates the approach by the authors in [9]. This is not the only possible choice, as several recent works improved some aspects of such approach, e.g., the one in [8]. However, consider that our scenario is highly dynamic, and the major source of energy consumption for UAVs is the activation of the RF interface for transmitting and receiving messages. Thus, we made such a choice so as to reduce the amount of over-the-air messages required to detect proximity among UAVs.

Two solutions that might appear close to ours are the ones by the authors in [25] and [30]. The proposal by [25], namely PPTM, considers the problem of finding any collisions in the overall path of two drones at runtime, thus focusing on a problem different from ours. The authors in [30], instead, proposed PPCA, a scheme for privacy-preserving collision avoidance, using a tessellation logic based on capsules, where both the source and the destination location of the moving entity are needed to compose the capsule. We compare LPPD against both [30] and [25], assuming two reference scenarios. The first case is the one of RPASs, i.e., the scenario assumed in this manuscript. Here, for all compared approaches, we assume that the next location to be travelled by the drone is known to allow us to create *capsules*, which is the geometrical shape needed by both PPCA and PPTM to be applied. We first consider the required number of modular exponentiations, multiplications, additions, and messages the devices involved in the protocol(s) exchanged. As PPCA runs on elliptic curves, we consider the equivalent number of operations required by such protocol to achieve the same result but on modular groups rather than elliptic curves. We summarize the results of our analysis in Tab. 5. For the RPAS scenario, LPPD is the protocol requiring the least exponentiations and the least number of messages exchanged between the devices to be aware of the co-location. The second case we consider is the case of fully autonomous vehicles. Such a scenario is the one for which PPTM has been designed; thus, such a protocol requires no assumptions. Instead, PPCA and LPPD have been designed for proximity detection on a single location. Thus,

Table 5: Complexity of LPPD, PPCA and PPTM, assuming the scenario of RPASs.

Solution	Exps.	Mults.	Adds.	Total Msgs.
PPCA [30]	5	5	0	3
PPTM [25]	3	0	0	3
LPPD	2	2	0	2

to adapt them to this case and compare them to PPTM, we need to assume that all locations travelled during the flight are known in advance. Assume that n is the number of locations travelled by the drone. The straightforward application of both PPCA and LPPD to the considered problem requires n times the overhead reported in Tab. 5, as the protocol should be executed completely for each couple of consecutive coordinates. Instead, the overhead required by PPTM depends on the presence (or not) of collisions among the drones and, overall, on the similarity among the two trajectories. In the best case, i.e., when the trajectories are very different, PPTM requires the same overhead reported in Tab. 5. In the worst case, i.e., when the two trajectories are the same, PPTM requires approx. $log_2(n)$ times the overhead reported in Tab. 5, being still the most convenient solution. Such a result is expected, as PPTM has been designed specifically to address the problem of trajectory matching, i.e., to verify the existence of any collisions among the trajectories. Conversely, our proposal LPPD has been designed for the case of co-location detection on the current location only, which is the only one known by RPASs, being more convenient for this latter case—exactly the problem considered in this manuscript. We also provide some quantitative numbers about the performance of the discussed protocols. As LPPD and the proposal in [30] use exactly the same hardware (3DR-Solo) and software (3DR-Poky), with the same software configuration, we can directly compare their performance. To estimate the performance of PPTM [25], we consider that, as reported in Tab. 5, PPTM requires three total messages exchanged between the entities and three modular exponentiations only, i.e., 2 exponentiations and 5 multiplications less than PPCA. We summarize in Tab. 6 the time, energy, and bandwidth requirements of both LPPD and the proposals in [30] and [25] for the case of RPASs. Assuming the reference case of the security level of 80 bits, PPCA [30] requires 72.01 mJ of energy per protocol run, while we estimate PPTM would require 61.824 mJ per run. LPPD, instead, requires only 14.15 mJ, i.e., ≈ 22.3 % of the time required by PPTM, originating partly from reduced computations and mostly from the limited bandwidth overhead. Instead, consider the case of fully autonomous vehicles and thus, full trajectories of $n = 50$ non-colliding coordinates (the best case of PPTM). We estimate PPTM would require exactly the same overhead (61.824 mJ and 3 messages), while LPPD would require n times the overhead reported in Tab. 6, i.e., 1, 983.15 mJ and 100 messages, being largely inefficient (more than 32 times the overhead of PPTM). We highlight, once again, that this result is fully expected, as our proposal has been conceived for the case of RPAS, while PPTM [25] has been conceived for fully autonomous systems.

At the same time, both LPPD and PPCA [30] allow choosing any collision avoidance strategy after the proximity warning is issued.

Table 6: Comparison between LPPD, [30] and [25] in the case of RPAS, assuming a common security level of 80 bits.

Solution	Time $[ms]$	Energy $[mJ]$	Bandwidth Overhead $[B]$	No. of Messages
PPCA [30]	97.310	72.009	3, 360	3
PPTM [25]	63.668	61.824	640	3
LPPD	39.663	14.15	299	2

A complete overview of all the approaches for proximity detection and avoidance on UAVs can be found in [38], while Tab. 4 reports a qualitative comparison between LPPD and the techniques described above, along significant system features.

We notice that privacy and security aspects for RPASs are hardly considered. The proposed solutions often rely on a Trusted Third Party (TTP) and assume that the UAVs are related to the same manufacturer. Compared to the discussed schemes, LPPD considers a scenario where the UAVs are previously unknown. Moreover, from Tab. 4, LPPD emerges as the only solution suitable for RPASs and capable of providing privacy-preserving proximity detection without relying on dedicated sensors on-board, and neither requiring the deployment of ad-hoc components at the ground. At the same time, the effectiveness of LPPD is also guaranteed when remote UAVs approach at high speed, where the reduced detection range might limit techniques integrating sensor readings. The combination of all the above features makes LPPD a compelling, unique, and novel solution for proximity detection on UAVs, enabling collision avoidance.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed LPPD, the first solution for lightweight privacy-preserving proximity discovery for remotely-piloted Unmanned Aerial Vehicle. LPPD integrates a novel space tessellation logic based on randomized spheres with a lightweight solution for private-set intersection, allowing two communicating UAVs to discover whether they are in proximity. We demonstrated LPPD to be fully reliable, guaranteeing a proximity detection ratio of 1 while also being viable and lightweight: an essential requirement for commercial UAVs, where optimal use of the energy budget is at a premium. Indeed, our performance assessment on the 3DR Solo drone demonstrated that, with the most lightweight but secure configuration, two UAVs can complete LPPD in only 39.66 milliseconds, while consuming only 14.15 mJ of energy, i.e., the $5 \cdot 10^{-6}$% of the battery capacity. The security of LPPD has been formally verified. Moreover, the security guarantees provided by LPPD can also be strengthened by selecting different configuration parameters while remaining efficient and viable for integration on modern UAVs and near-real-time proximity detection. Future work includes the extension of LPPD in a broadcast scenario.

ACKNOWLEDGMENTS

The authors thank the anonymous Reviewers for the constructive comments on the paper. This publication was partially supported by Technology Innovation Institute, Abu Dhabi, EAU and by NATO MYP G5828 project "SeaSec: DronNets for Maritime Border and Port Security". This work has been partially supported also by the INTERSECT project, Grant No. NWA.1162.18.301, funded by the Netherlands Organisation for Scientific Research (NWO). The findings reported herein are solely responsibility of the authors.

REFERENCES

[1] 2019. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access* 7 (2019), 87658–87680.
[2] Coluccia A., et al. 2014. RSS-Based Localization via Bayesian Ranging and Iterative Least Squares Positioning. *IEEE Commun. Let.* 18, 5 (2014), 873–876.
[3] 3D Robotics. 2020. Yocto Linux. https://tinyurl.com/y2axm74b. (Accessed: 2023-Sep-29).

[4] 3DR Robotics. 2014. MAVlink Protocol Setup for Solo. https://github.com/3drobotics/mavlink-solo. (Accessed: 2023-Sep-29).
[5] 3DR Solo Website. 2020. 3DR Solo Website. https://3dr.com/solo-drone. Accessed: 2023-Sep-29.
[6] E. Barker. 2020. *NIST Special Publication 800-57 Part 1 – Revision 5 - Recommendation for key management: Part 1.* Technical Report.
[7] B. Blanchet. 2014. *Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif.* 54–87.
[8] Geoffroy Couteau. 2018. New Protocols for Secure Equality Test and Comparison. In *Proc. of Applied Cryptography and Network Security.* Springer, 303–320.
[9] E. De Cristofaro and G. Tsudik. 2010. Practical Private Set Intersection Protocols with Linear Complexity. In *Financial Cryptography and Data Security.* 143–159.
[10] DJI. 2019. DJI Demonstrates Direct Drone-To-Phone Remote Identification. https://tinyurl.com/y39pft7x. (Accessed: 2023-Sep-29).
[11] FAA. 2021. Remote Identification of Unmanned Aircraft. Available Online: https://www.faa.gov/news/media/attachments/RemoteID_Final_Rule.pdf.
[12] J. Gelinas. 2019. Look, up in the sky! It's my package. Amazon to start drone delivery 'within months'. https://tinyurl.com/yrnj9xtj
[13] Y. Hsu and R. Gau. 2020. Reinforcement Learning-based Collision Avoidance and Optimal Trajectory Planning in UAV Communication Networks. *IEEE Trans. on Mobile Computing* (2020), 1–1.
[14] Huang, S. et al. 2019. Collision avoidance of multi unmanned aerial vehicles: A review. *Annual Reviews in Control* 48 (2019), 147–164.
[15] J. Park, et al. 2008. UAV collision avoidance based on geometric approach. In *SICE Annual Conf.* 2122–2126.
[16] Keranidis, S. et al. 2014. Experimental Evaluation and Comparative Study on Energy Efficiency of the Evolving IEEE 802.11 Standards. In *Proc. of Int. Conf. on Future Energy Systems.* 109–119.
[17] Kobeissi, N. et al. 2019. Noise Explorer: Fully Automated Modeling and Verification for Arbitrary Noise Protocols. In *IEEE EuroS&P.* 356–370.
[18] Ma, Y. et al. 2019. WiFi Sensing with Channel State Information: A Survey. *ACM Comput. Surv.* 52, 3, Article 46 (June 2019), 36 pages.
[19] N. Gageik, et al. 2015. Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors. *IEEE Access* 3 (2015), 599–609.
[20] Oligeri, G. and Sciancalepore, S. and Raponi, S. and Di Pietro, R. 2020. BrokenStrokes: on the (in) security of wireless keyboards. In *Proc. of ACM WiSec.* 231–241.
[21] OpenSSL Found. 2021. OpenSSL - Cryptography and SSL/TLS Toolbox. https://www.openssl.org/. (Accessed: 2023-Sep-29).
[22] R. Beard, et al. 2003. Multiple UAV cooperative search under collision avoidance and limited range communication constraints. In *IEEE Int. Conf. on Decision and Control*, Vol. 1. 25–30.
[23] Fabio Ricciato, Savio Sciancalepore, Francesco Gringoli, Nicolò Facchi, and Gennaro Boggia. 2018. Position and Velocity Estimation of a Non-Cooperative Source From Asynchronous Packet Arrival Time Measurements. *IEEE Transactions on Mobile Computing* 17, 9 (2018), 2166–2179.
[24] Bahareh Sabetghadam, Rita Cunha, and António Pascoal. 2022. A Distributed Algorithm for Real-Time Multi-Drone Collision-Free Trajectory Replanning. *Sensors* 22, 5 (2022), 1855.
[25] Sciancalepore, S. and George, D. 2022. Privacy-Preserving Trajectory Matching on Autonomous Unmanned Aerial Vehicles. In *ACM ACSAC.*
[26] Sharma, A., et al. 2020. Communication and networking technologies for UAVs: A survey. *Jour. of Netw. and Comput. Applic.* 168 (2020), 102739.
[27] Shen, K. et al. 2022. Multidepot Drone Path Planning With Collision Avoidance. *IEEE IoT Journ.* 9, 17 (2022), 16297–16307. https://doi.org/10.1109/JIOT.2022.3151791
[28] Pietro Tedeschi, Fatima Ali Al Nuaimi, Ali Ismail Awad, and Enrico Natalizio. 2023. Privacy-Aware Remote Identification for Unmanned Aerial Vehicles: Current Solutions, Potential Threats, and Future Directions. *IEEE Transactions on Industrial Informatics* (2023), 1–12.
[29] Tedeschi, P. and Sciancalepore, S. and Di Pietro, R. 2021. ARID: Anonymous Remote IDentification of Unmanned Aerial Vehicles. In *ACM ACSAC.*
[30] Tedeschi, P. and Sciancalepore, S. and Di Pietro, R. 2022. PPCA - Privacy-Preserving Collision Avoidance for Autonomous Unmanned Aerial Vehicles. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–1.
[31] Tedeschi, P. and Sciancalepore, S. and Di Pietro, R. 2022. Source code of LPPD in ProVerif. https://github.com/pietrotedeschi/lppd.
[32] Tedeschi, P. and Sciancalepore, S. and Eliyan, A. and Di Pietro, R. 2020. LiKe: Lightweight Certificateless Key Agreement for Secure IoT Communications. *IEEE IoT Journ.* 7, 1 (2020), 621–638.
[33] Unmanned Airspace. 2019. "526,000 commercial drones to be delivered in 2020". https://tinyurl.com/mr26xmsy. Accessed: 2023-Sep-29.
[34] M. Vanhoef and F. Piessens. 2017. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proc. of ACM CCS.* 1313–1328.
[35] Eva Wisse, Pietro Tedeschi, Savio Sciancalepore, and Roberto Di Pietro. 2023. A^{2}RID—Anonymous Direct Authentication and Remote Identification of Commercial Drones. *IEEE Internet of Things Journal* 10, 12 (2023), 10587–10604.
[36] X. Wang, et al. 2007. Cooperative UAV Formation Flying With Obstacle/Collision Avoidance. *IEEE Transactions on Control Systems Technology* 15, 4 (2007), 672–679.
[37] Y. Watanabe, et al. 2007. Vision-Based Obstacle Avoidance for UAVs. In *AIAA Guidance, Navigation and Control Conf. and Exh.* 6829.
[38] Yasin, J., et al. 2020. Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches. *IEEE Access* 8 (2020), 105139–105155.

ANNEX: NOTATION TABLE

Table 7: Notation used throughout the paper.

Notation	Description
u_i	Generic UAV, with N number of UAVs.
n_i,	RSA modulus selected for the protocol.
e_i, d_i	Public and private portions of the key of u_i.
g	Group generator.
p_i, q_i	Large primes of u_i, where $q_i = k(p_i - 1)$.
$\mathsf{pos_i}$	GPS location of UAV u_i.
(lat_i, lon_i, alt_i)	GPS coordinates of UAV u_i.
$\tilde{x_i}, \tilde{y_i}, \tilde{z_i}$	Single location Identifiers of UAV u_i.
$p\mathcal{G}s_i$	Plaintext location identifier of UAV u_i.
M_k	Manufacturer of u_i, releasing the public key certificate C_i.
O_i	Origin selected by u_i for the tessellation.
δ	Maximum GPS error on the location of u_i.
r_i	Sphere radius chosen by UAV u_i.
r_α	Sphere radius chosen by UAV α.
h_i	Position digest generated by UAV u_i.
t_i	Tag digest generated by UAV u_i.
ψ_i	Ciphertext generated by UAV u_i.
K_i	Parameter of LPPD generated by UAV u_i.
ρ_i	Random value generated in $\mathsf{Z}^*_{n_i}$ from UAV u_i.
C_i	Public-key certificate of u_i.
B	Operation bandwidth of the UAV u_i.
$\mathcal{A}$	Adversary.
$\mathcal{H}$	Hash function.
ts	Timestamp.
T_i	Threshold distance selected by the UAV u_i.
$d_{\alpha,j}$	Distance from UAV i and UAV j.
$\mathsf{s} = s_x, s_y, s_z$	Random nonces used to generate a random origin point of the space tessellation.
S	Symmetric encryption algorithm.
E	Public-key encryption algorithm.
D	Public-key decryption algorithm.