Strangers Sets: Preserving Drones' Location Privacy while Avoiding Violations of Critical Infrastructures

Alessandro Brighente
University of Padova
Padova, Italy
alessandro.brighente@unipd.it

Mauro Conti
University of Padova
Padova, Italy
mauro.conti@unipd.it

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

Harshul Vaishnav
University of Padova
Padova, Italy
harshul.vaishnav@studenti.unipd.it

ABSTRACT

Preserving the location privacy of drones while allowing Critical Infrastructures (CIs) to identify nearby drones and their violations represents a significant challenge. To allow for improved accountability of drone operations, the current standard by the Federal Aviation Administration (FAA) mandates drones to disclose their location (in cleartext). However, such a strategy provides malicious eavesdroppers with significant possibilities for tracking and profiling, thus jeopardizing users' privacy. A recent proposal suggested using geo-indistinguishability to sanitize drones' locations while allowing CIs to detect violations. However, due to the statistical nature of the approach, the risk of false violation detection is inversely proportional to the privacy guarantees of drones.

In this paper, we propose Privacy Preserving vIolation Detection (PPID), a novel approach using a private set intersection algorithm to simultaneously protect drones' and CIs' location privacy and allow CIs to detect violations while avoiding the problem of false violation detection. We propose two versions of the protocol: i) PPID, which uses an elliptic curve-based private set intersection inspired by relevant literature to detect the co-presence of drone and CI in a given area in a privacy-preserving fashion, and ii) extended (e)-PPID, which extends the message with an approximation of the future location of the drone, to predict possible future violations. To validate our proposal, we implement our protocols and deployed them on a proof of concept involving resource-constrained devices. We extract performance metrics regarding the security, execution time, communication cost, and memory overhead incurred by our protocols. Our results show that PPID and e-PPID provide accurate results about a violation requiring approx. 52ms and 84ms, respectively, in the worst-case scenario (i.e., the highest possible number of messages exchanged) and for a 256-bit security level.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; **Public key encryption**; **Digital signatures**; **Pseudonymity, anonymity and untraceability**.

SAC '25, March 31-April 4, 2025, Catania, Italy

KEYWORDS

Privacy Enhancing Technologies, Unmanned Aerial Vehicles, Embedded Systems Security; Network Performance Assessment.

ACM Reference Format:
Alessandro Brighente, Mauro Conti, Savio Sciancalepore, and Harshul Vaishnav. 2025. Strangers Sets: Preserving Drones' Location Privacy while Avoiding Violations of Critical Infrastructures. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3672608.3707724

1 INTRODUCTION

The availability of Unmanned Aircrafts (UAs) (a.k.a. drones[1]) on a commercial scale, as well as the possibility to equip them with Information Technology (IT)-based modules, paved the way to the investigation of cyber-security and privacy for such mobile devices. Indeed, drones can be victims of cybersecurity attacks or deliverers of security and safety-threatening actions. When victims of cyberattacks, the value at stake is generally either the hardware of the drone itself, i.e., the drone and its payload, or the data generated, processed and stored onboard. In the first case, the attacker exploits the hardware components of the drone to undermine its correct functioning and possibly deactivate it [1, 2]. In the second case, the attacker either targets the information that a drone may collect and process in different use cases [3, 4], or aims at tracking and profiling the drone [5, 6]. When drones are used for threatening actions, famous examples include the attack to the Gatwick airport [7] or the attacks carried out in the US and Saudi Arabia in 2021 [8]. Indeed, drones can be a safety threat by solely occupying reserved and critical airspaces. Furthermore, drones can be equipped with recording devices (e.g., cameras and microphones) to snoop sensitive information. Therefore, it is fundamental to protect the airspace from unauthorized access by UAs [9]. To this aim, sensitive targets such as Critical Infrastructures (CIs) should be equipped with drone detection technologies. To support the accountability of UAs operations, the Federal Aviation Administration (FAA) mandates the use of *Remote ID* [10], a rule that requires drones to periodically broadcast clear-text information such as the identification number, location of the UA, and location of the controller. Although useful for identification, Remote ID requires the disclosure of UA's sensitive information. Despite not being the intended scope, the indiscriminate

[1] In this paper, we use the term UA and drone interchangeably.

broadcast of the information previously cited undermines drones' privacy. Since its proposal, many drone users and manufacturers raised complaints about the lack of privacy of Remote ID, resulting in more than 53k comments and a lawsuit [11] against the FAA. Therefore, *Remote ID* fully sacrifices drones' location privacy on the altar of making drones operations fully accountable.

Motivation. Protecting the location privacy of drones and providing means to CIs to detect the presence of intruders should not be a trade-off. Indeed, it is fundamental for drone users, especially for the ones performing commercial and sensitive military operations, to be able to fly drones in public spaces without necessarily disclosing uncontrollably sensitive information that may lead to tracking and profiling. At the same time, CIs need to be able to detect the presence of intruding drones, to protect their assets and avoid safety issues (e.g., a drone dropping a bomb over a nuclear plant area). Therefore, CIs should receive location information from UAs flying close to the CI's area. The only solution to the problem currently available in the literature proposes the use of differential privacy to sanitize the drone's location information while providing means to the CI to detect the presence of a UA in a no-fly area [12]. Although this represents a reasonably practical solution (e.g., no need to share keys between CI and UA), it is prone to false positives due to its statistical approach. Indeed, based on the differential privacy parameters, the authors showed a trade-off between the privacy level achieved by the drone and the performance of the CI in detecting invaders. As violation detection requires the CI to take actions to defend against an intruder, an effective detection approach requires minimal/no false positive detection. However, such a solution is currently not available in the literature.

Contributions. In this paper, we propose Privacy Preserving vIolation Detection (PPID), an innovative solution to simultaneously protect drones and CIs' location privacy and allow CIs to detect violations from UAs. Our solution is based on *private set intersection*, i.e., a cryptographic construct that allows for the detection of intersections over sets without disclosing sensitive information. Thanks to such a cryptographic approach, we avoid the problem of false violation detection, without sacrificing the location privacy guarantee of the UAs. We first propose our protocol and describe the information that UAs and the CI should exchange. Then, we propose Extended-Privacy Preserving vIolation Detection (e-PPID), an extended version of our protocol based on the predictability of the location of a UA in successive time frames. To validate the feasibility of our solutions, we implement them on a resource-constrained device featuring resources similar to the ones of a modern drone, and we evaluate their execution time and memory requirements for different protocol configurations. Our results show that the proposed protocols can run on resource-constrained devices with an average runtime of tens of milliseconds, thus representing a suitable solution for fast violation detection. Furthermore, they can run with limited communication costs (kByte order) and low memory overhead (hundreds of kByte order), being hence suitable for implementation on a resource-constrained device such as drones.

We summarize our contributions as follows.

- We propose PPID, a novel protocol that allows a CI to detect a violation from a UA while preserving the location privacy of both the UA and the CI. Our solution provides means for the CI to notify the UA pilot about the violation to have the UA change its course.
- We propose an extended version of PPID, i.e., e-PPID, that aims at preventing future violations. This solution envisions the prediction of the UA's future location based on its current direction and speed. By including the encrypted future location in the communication with the CI, the UA allows the CI to predict whether the UA is going to invade a no-fly area, and warn the pilot beforehand.
- We evaluate our protocols at different levels to show their feasibility. We first assess their security against possible attacks by using the automated verification tool *ProVerif*. We then implement them on a resource-constrained device to emulate their execution on a commercial drone. Our results show that PPID and e-PPID provide accurate results about a violation requiring approx. 52ms and 84ms in the worst-case scenario, respectively, for a 256-bit security level.

Overall, our solution can be applied whenever RID regulations do not apply, e.g., due to the limited weight of the drone, sensitive nature of the operations (e.g., military domain) or in specific FAA-Recognized Identification Areas (FRIAs), not requiring compliance with RID regulations.

Organization. The rest of the paper is organized as follows. Sec. 2 introduces the system and threat models, Sec. 3 describes the proposed protocols, Sec. 4 provides an extensive evaluation of our solutions, Sec. 5 compares our solutions to the current state of the art and, finally, Sec. 6 concludes the paper and outlines future work.

2 SYSTEM AND THREAT MODEL

We provide here a description of the entities and system model considered in our work. In particular, we describe the system model in Section 2.1 and the threat model in Section 2.2.

2.1 System Model

We consider a scenario where a CI needs to regulate the physical access to its proximity due to safety and privacy concerns. The CI operator would like to identify violations of the monitored area, namely the *no-fly area*, by unauthorized UAs, to prevent eavesdropping of sensitive information both in terms of audio and video recordings from suitably equipped UAs. To this aim, the CI periodically emits beacon packets, that can be used by surrounding drones to demonstrate that they are not invading the no-fly area.

On the UA side, we consider drones are equipped with wireless communication capabilities compatible with the ones used by the CI, i.e., they share the same wireless communication technology. Thus, the UA can monitor the presence of messages originating from a CI and possibly reply with their own messages. In particular, upon receiving a beacon, the drone can reply to the CI (we provide more details in Sec. 3). Also, we consider drones featuring a GPS receiver, so as to be able to compute their own actual location, in terms of latitude, longitude, and altitude. We believe such considerations align with the equipment available on most drones used outdoor for long-range missions, especially taking into account the mandatory adoption of RID capabilities by all commercial drones starting from 2024 in the US and EU [13]. We consider drones (and pilots) that behave honestly. Thus, they do not spoof their location and hence provide their actual current location. Although this assumption might not hold for highly skilled malicious users aiming at jeopardizing the CI's safety and privacy, we highlight that, by design, it is not possible to provide means for non-malicious UAs to protect their location privacy while avoiding invading a no-fly area. Thus, our proposal provides a solution to a reasonable problem.

To support the decision on the violation by providing precise information on the drone's location, the CI can rely on the Unmanned Service Supplier (USS), i.e., a trusted service able to disclose the true location of the UA. Once the CI detects a possible violation, it forwards proof of such violation to the USS. The USS can then obtain the actual drone location, verify the occurrence of a possible violation and, if this is actually happening, report the actual location back to the CI. In the protocol description in Section 3, we discuss possible actions that the CI might undertake upon confirming the violation.

At first glance, a naive solution would be for the USS to continuously check drone messages looking for violations of no-fly areas. However, such a method would cause excessive overhead on the USS. Considering that the USS is likely deployed and managed by government entities, such overhead would generate concerns about its deployment, causing such a naive solution to never become a reality. In our setup, it would be also unfeasible for the CI to broadcast messages including the boundaries of the NFZ in plaintext. Such boundaries constitute sensitive information, that can never be disclosed to potentially untrusted parties listening on the wireless channel.

2.2 Threat Model

In this paper, we consider two threats: (i) an unaware pilot unintentionally flying his drone close to a CI area, and (ii) a passive attacker trying to track a drone, as described below.

Unaware pilot, $\mathcal{A}_1$. This entity is an amateur drone pilot whose drone may unintentionally approach the CI area, risking safety and potential damage from anti-drone technologies. We assume $\mathcal{A}_1$ cannot tamper with the UA firmware or send false location data, as our model focuses on compliant users rather than malicious pilots. Since the CI cannot differentiate between malicious and unaware drones, it takes countermeasures upon detecting a violation. Therefore, we assume $\mathcal{A}_1$ moves away from the CI area after receiving a warning.

Passive attacker, $\mathcal{A}_2$. We assume that the objective of $\mathcal{A}_2$ is to track a victim UA. Note that attackers can exploit this process for multiple purposes. For instance, due to the unmanned nature of UAs, an attacker able to track a delivery UA may physically capture it and steal the carried payload. Furthermore, the leakage of the location of the UA may represent a threat also due to those that are "enraged by drones" [10]. Lastly, a company may want to track UAs belonging to a competitor to cause financial damages. The feasibility of this attack is currently confirmed by the communication model of *RemoteID*, which mandates drones to broadcast sensitive information such as identifier and location in clear [14]. Hence, the attacker may be able to track a target UA thanks to the presence of the unique UA identifier contained in each *RemoteID* message.

Considerations on Stronger Attackers. When considering malicious and skilled attackers able to modify the firmware of the drone so as not to transmit actual location information, schemes like ours based on a higher layer (with reference to the ISO/OSI model) cannot reliably guarantee violation detection. Feasible solutions to detect such misbehavior might be based on the time difference of arrival for location estimation [15]. However, we point out that such an attacker is outside the scope of our work, which aims at providing a way for honest pilots to preserve their privacy while avoiding entering no-fly areas.

Considerations on CIs Location Privacy. We note that our protocol also avoids direct disclosure of the location of the CI, besides the one of the UA. However, opposite to UAs, consider that: (i) the location of the CI does not change over time, and (ii) the CI continuously broadcasts beacons. Thus, based on the characteristics of the scenario, attackers may attempt to localize the CI using traditional wireless localization techniques, based either on the usage of the Received Signal Strength (RSS) of the packets or their Time-of-Arrival (ToA) [16], [15]. To preserve its location privacy against malicious physical layer-based localization, the CI should additionally employ techniques such as jamming [17], the creation of fake locations [18], or a relay system [19]. Alternatively, the CI can employ a set of directional antennas each with a different transmission range. This would break the basic assumption of localization algorithms, i.e., that the transmitter is at the center of the localization area, making precise localization unfeasible. Conversely, due to the dynamic nature of the UAs and their opportunistic interaction through the protocol, such localization techniques hardly apply to drone localization.

3 PROTOCOL DESCRIPTION

In this section, we describe our proposed protocols. We define the location encoding process in Section 3.1 and describe PPID in a nutshell in Section 3.2. Sections 3.3, 3.4, and 3.5 describe the phases required by PPID. We then introduce e-PPID in Section 3.6.

3.1 Location Encoding

Similarly to [20], PPID is based on a tessellation logic aimed at detecting proximity between UA and CI. To this aim, we leverage Location Encoding[2] to encode geographic coordinates into points over a tasselled space. In particular, given a circular tessellation, we define $d_{\alpha,j}$ as the maximum allowed distance between the entity α and any other entity j, and we set the origin of the tessellation $\mathbf{O} = [O_x, O_y, O_z]$ according to $O_x = x_n - s_x \cdot d_{\alpha,j}; O_y = y_n - s_y \cdot d_{\alpha,j}; O_z = z_n - s_z \cdot d_{\alpha,j}$; where $s_x, s_y, s_z \in \mathcal{N}$ are random nonces and x_n, y_n, z_n are the current x, y, and z coordinates of the drone. We then use Cantor pairing [21] to map the center coordinates to the *mapped location*, i.e., a scalar integer value. Note that such a strategy is required for two reasons. First, by choosing the center of the tessellation based on the current location of the drone, we avoid issues related to locations at the boundaries between different circles, which would happen with a static tessellation, instead. Second, by randomizing the initial point of the space tessellation, we prevent the adversary ϵ from pre-computing the mapping between locations and tessel identifiers, further enforcing location privacy. At the same time, two points at a distance closer than $d_{\alpha,j}$ will be mapped into the same location by the Cantor pairing, leading to co-location detection in our protocol.

Note that such an encoding process based on circles cannot map all possible coordinates, as circles do not perfectly adhere to one another. This introduces gaps in the mapping, namely, no-mapping zones[3], i.e., locations lying in no available circle. Recall that the aim of our solution is to detect co-location; thus, whenever the drone maps its actual location to a no-mapping zone, it can be sure that there is no co-location.

Finally, note that, for some particular locations (e.g., closer to poles) circles might become shrank and stretched when mapped on the WGS84 ellipsoid modelling the Earth's surface. This is not an

[2]Notice that, in this paper, we use both terms *location mapping* and *location encoding* as synonyms.

[3]no-mapping zone and no-fly area are different terms. No-fly area refers to a restricted area whereas no-mapping zone corresponds to a coordinate that does not lie in any circle.

issue for the effectiveness of the protocol, as the distance covered by a mobile device in practice still remains the same.

3.2 PPID in a Nutshell

In a nutshell, our solution, namely, PPID, involves the delivery of broadcast messages from both *active observers* installed and managed by CI operators. Our solution is partly inspired by the protocol proposed by Narayanan et al. in [22]. Due to the limited space available, we omit the description of the reference protocol and rather describe the protocol flow of our solution (readers can see Sec. 5 for an overview of our contributions compared to the protocol originally proposed in [22]).

Overall, we envision three phases, namely, the *Setup*, *Runtime*, and *Disclosure* phases. In the *Setup Phase*, executed offline, both the CI and UAs' operators set up the protocol by registering identification information with the USS, and receiving public parameters.

As detailed in Sec. 3.4, at runtime (e.g. every 1 sec as recommended by Remote ID, the UA and the CI apply a Private-Set Intersection protocol over a broadcast wireless channel to check for co-location. The active observers of the CI operators periodically broadcast wireless messages (beacons), including a challenge, i.e., a function of their location and no-fly area, their public key and related certificate, a timestamp, origin, and radius of the no-fly area, and a signature (to ensure authenticity). At message reception, any UAs flying in the neighborhoods of the active observers, upon message authenticity and validity verification, encodes its own location in the space tessellation logic provided by the active observers, computes a response to the received challenge by using such *mapped location*, and delivers such an encrypted response in the next message.

Whenever a message is received by a CI, the observers first verify the authenticity and freshness of the received message. Then, they look for the presence of any responses to their challenges and perform a comparison in the encrypted domain. If the result of the operation is a *match*, it means that the UA is flying within the CI's no-fly zone, and thus, a *violation* is ongoing. As detailed in Sec. 3.5, to disclose the actual location of the UA and take countermeasures, the observers forward the received messages to the USS, i.e., the only entity able to unveil the actual location of the UA. The USS verifies the authenticity and freshness of the received message, as well as the violation of the no-fly area of the reported entity. In case of violation, it unveils to the observer the location of the UA. The USS can also take further (legal) actions against the intruder, using the authentic message reported by the observer. We report more details on all the steps and phases below.

We provide the details of all such phases in the following subsections.

3.3 Setup Phase

Figure 1 shows the steps executed by the involved entities during the setup phase. Note that the UA and the CIs all communicate to the USS. Thus, no mutual knowledge is required. The setup phase is executed offline by the CI, UA, and USS. The CI registers its public information with the USS, including CI's (observers) actual location, no-fly area, and certificates. The UA registers its certificate $Cert_n$ and its ID D_n to the USS. In response, the USS provides the settings and parameters of the protocol to both the CI and UA. Such parameters include the prime number p, curve parameters z_1 and z_2, the cyclic group Z, the generator G, and the order n of the group. The UA receives a number L of ephemeral public/private keys from the USS. The drone uses

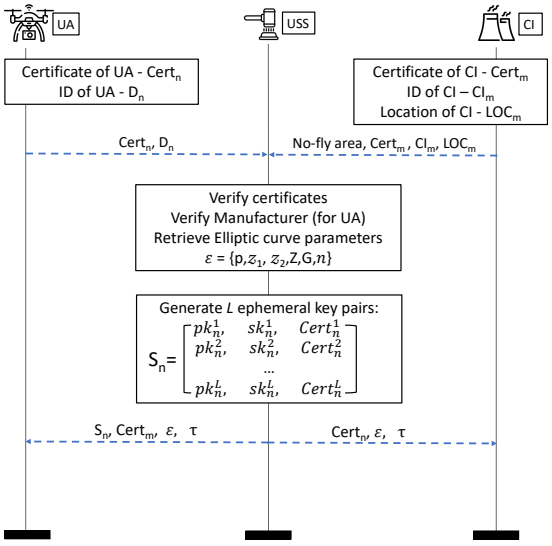


Figure 1: Sequence diagram of the setup phase of PPID.

these pairs to sign its messages in a pseudo-anonymous manner. Since the messages of a drone can be overheard only for some kilometres, the drone may also re-use keys throughout the mission at different locations, likely not compromising its anonymity. For each UA, the USS generates unique key pairs. The generation of these ephemeral keys is out-of-scope of this paper. However, it can be done through traditional ECC key generation techniques.

3.4 Runtime Phase

Figure 2 shows the sequence diagram of the runtime and disclosure phase of the protocol. For ease of notation, we denote the CI as entity A (initiator) and UA as entity B (responder). The protocol can be subdivided into 3 runtime sub-phases. In the following, we discuss each phase in depth.

Beacon Generation. The overall protocol is initiated by the CI. The CI executes the protocol to detect whether any UAs in the surroundings are invading its no-fly area. To this aim, the CI broadcasts a beacon message for the UAs in the surrounding area.

The CI first calculates the timestamp t_A, used to check the validity of the challenge message. It also extracts a K bits nonce μ_A, which is used to encrypt CI's mapped location. The CI maps its actual location through the parameters provided to the USS, i.e., the origin, O_A and radius, r_A of the circle obtained by the tessellation of Earth's surface, as described in Section 3.1. The mapped location loc_A is encrypted through the CI's public key using El-Gamal elliptic curve cryptography as, in Eq. 1.

$$C_A = (C_{A,1}, C_{A,2}) = \left(\mu_A G, (loc_A + \mu_A) pk_A\right). \quad (1)$$

The CI then signs the message using its private key, according to Eq. 2, where *sign* denotes a generic public-key signature algorithm.

$$\delta_A = sign\left[H\left(C_A, t_A, O_A, r_A, Cert_A\right), sk_A\right]. \quad (2)$$

Finally, the CI assembles and broadcasts a beacon containing the challenge C_A, signature δ_A, timestamp t_A, public key certificate, O_A, and r_A of the no-fly area.

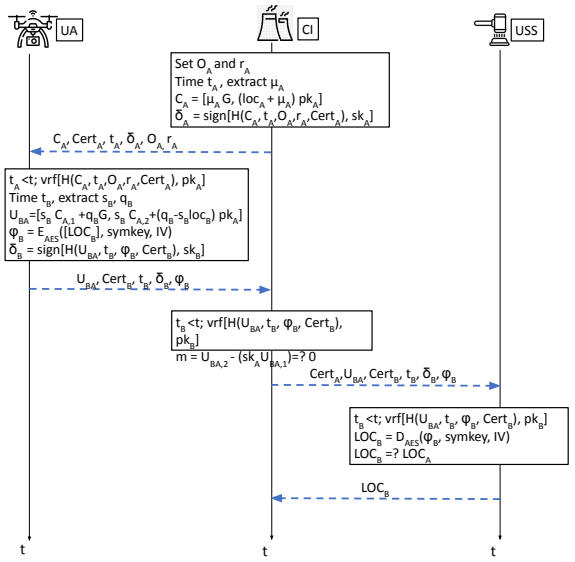


Figure 2: Sequence diagram of PPID.

Beacon Verification and Response Generation. Upon receiving the beacon, the UA first checks if the CI is certified by the USS. The UA stops the protocol if the certificate is not valid. Otherwise, it continues the protocol to verify the integrity and validity of the received message using the public key of CI via (3), where vrf denotes a generic public-key signature verification algorithm.

$$vrf\left[\delta_A, pk_A\right] = H\left(C_A, t_A, O_A, r_A, Cert_A\right). \tag{3}$$

If the message is verified and received within the time limit, the UA continues with the protocol; otherwise, it drops the message and stops protocol execution. Note that the check on the time validity of the message is only required to detect (and reject) CI spoofing attacks. In regular operational conditions, such a check is always satisfied.

Once all verification checks have been successfully completed, the UA generates the response $U_{BA} = (U_{BA,1}, U_{BA,2})$ to the challenge contained in the beacon. Firstly, the UA computes the timestamp t_B for the validity of the response message. Then, it maps its actual location with the parameters provided by the CI, i.e., the O_A and r_A of the circle, to obtain loc_B. After mapping, it extracts two nonces of K bits, q_B and s_B, which are used for the computation of $U_{BA,1}$ and $U_{BA,2}$, as in (4).

$$U_{BA,1} = s_B C_{A,1} + q_B G, \quad U_{BA,2} = s_B C_{A,2} + (q_B - s_B loc_B) pk_A. \tag{4}$$

The UA appends the encryption of its actual location to the response. Moreover, the UA computes the *encrypted location report*, used to allow the USS to obtain the actual location of the UA in case of misbehavior. To encrypt the actual location of the UA, we use a two-party communication scheme, namely Elliptic Curve Integrated Encryption Scheme (ECIES) [23]. Both parties generate a shared secret key and Initialization Vector (IV), using their respective private key and the public key of the other party. This specific cryptographic technique is commonly referred to as the Elliptic-curve Diffie–Hellman scheme (ECDH) [24]. This shared key is used to encrypt the message symmetrically. Any valid ECC key pair can be used for ECDH. Thus, any ECC key pair can be used in our protocol, without requiring the generation of additional keys. Also, we do not need to share any symmetric keys, because the drone uses ephemeral keys and generates every time a new secret key. For symmetric encryption and decryption, we used the AES cryptosystem with a 128 bit key.

To generate the encrypted location report, the UA performs the following computations: (i) converts the coordinates into a single string; (ii) generates the ephemeral secret key, $symkey$ and IV using its ephemeral private key and the public key of USS; and (iii) encrypts the string containing the actual location as in (5) via the AES cryptosystem using the above-generated key and IV.

$$\phi_B = E_{AES}([loc_B], symkey, IV). \tag{5}$$

The value ϕ_B is appended to the response message. Finally, the response message is digitally signed according to (6) with the same ephemeral private key used for the shared key generation. The response message is now broadcast by the UA. It is worth noting that our proposal does not require any additional key to maintain or share.

$$\delta_B = sign\left[H\left(U_{BA}, t_B, \phi_B, Cert_B\right), sk_B\right]. \tag{6}$$

Response Verification and Violation Detection. Upon receiving a UA-generated response message, the CI checks if the UA is certified by the USS. The CI stops the protocol if the certificate is not valid. Otherwise, it continues the protocol to verify the integrity and the validity of the message using the public key of UA, as in Eq. (7).

$$vrf\left[\delta_B, pk_B\right] = H\left(U_{BA}, t_B, \phi_B, Cert_B\right). \tag{7}$$

If the message is verified and received within the time limit, then the CI continues with the protocol. Otherwise, it reports the issue to the USS and restarts the protocol.

After verification, the CI computes the value m according to (8).

$$m = U_{BA,2} - (sk_A \cdot U_{BA,1}). \tag{8}$$

If $m = O$, corresponding to the infinity point of the elliptic curve, then $loc_A = loc_B$ and the UA and CI lie in the same circle, leading to a violation. If $m \neq O$, then $loc_A \neq loc_B$, i.e., the UA and CI lie in two different circles and hence there is no violation. As the violation detection is performed in the encrypted domain, the CI cannot obtain any information on UA's location. If $m \neq O$, the CI stops the protocol since the UA is not invading its no-fly area. Otherwise, the CI forwards the response received from the UA to the USS.

3.5 Disclosure Phase

If the CI detects a violation, it communicates it to the USS and forwards the response message received from the UA. Upon receiving the violation request from the CI, the USS verifies the integrity of the message using the public key of the UA via Eq.(7), along with the validity of the message with the maximum validity time τ. If the message is verified and received within the time limit, the USS executes the remaining part of the protocol. Otherwise, it stops the protocol and responds to the CI with the verification failure message.

On successful verification checks, the USS decrypts the actual location of UA with the shared secret key, the IV generated using its private key, and UA's public key, using Eq. 9.

$$loc_B = D_{AES}(\phi_B, symkey, IV). \tag{9}$$

Then, it checks whether the actual location of the UA lies within the no-fly area of the CI. If so, it means that the drone is invading the CI. Thus, the USS reports to the CI the actual location of UA. If the USS determines that the UA is not invading the no-fly area of CI, then it responds with a false violation message.

If the UA is invading, the CI can choose to either disarm the UA or alert it. We envision the two following scenarios.

(1) The CI sets the radius for the protocol larger than its no-fly area boundary, creating an alert area. In this case, when the USS confirms a violation, the CI can decide to disarm or alert the drone.
(2) The USS creates a small session between the invading UA and the CI. This allows the UA to share its location with the CI, which allows the CI to track the UA within its territory continuously. As an example, the USS can use a time-based session, after which the drone will stop direct communication with the CI. It can be again extended with another proof of violation executed by the CI.

3.6 Extended-PPID protocol

In this section, we propose an extended version of PPID, namely, e-PPID, allowing the CI to avoid future violations by predicting the drone's future location based on its maximum speed and current direction (angle) of movement. The predicted location allows the CI to alert the drone before a violation occurs to avoid it.

The e-PPID protocol requires the UA to send an additional message, containing parameters related to its next location. Considering the current time instant t_0, we assume the drone predicts its next location as time $t_1 = t_0 + \Delta_t$, with Δ_t being a suitably selected parameter. If the UA remains in the same circle in both actual and future locations, then sending an additional message would not provide additional information and can thus be avoided. Therefore, this additional message is only sent if the drone predicts to move out of the current circle in the successive step.

To run the e-PPID, the UA does not need any additional information other than its maximum speed V_{max} and the current heading (direction or angle of movement). The assumption here is that the UA flies in a straight path. Since speed and direction are always known to the UA, it requires no additional computation. The drone B first computes the future location $LOC_{F'}$. It then checks whether the mapped actual and future location lies in the same circle, a different circle, or in a no-mapping zone. The UA executes this process such that it knows whether it will move out of the current circle or not, and therefore, whether to send one or two location reports.

Figure 3 shows the steps of e-PPID. In the following, we describe the protocol phases of e-PPID. e-PPID leverages the same steps of PPID in the setup phase. As for the runtime phase, the beaconing phase run by the CI stays the same. In the Beacon verification and response generation phase, after verification, the UA generates U_{BA} as in PPID. Then, assuming all the verification checks are passed, and the UA is moving out of the circle, it generates $F_{BA} = (F_{BA,1}, F_{BA,2})$. To this aim, the UA first generates two new nonces $q_{B'}$ and $s_{B'}$, which are used to compute F_{BA} along with $loc_{B'}$, i.e., the mapped predicted future location, as per Eq. 10.

$$F_{BA,1} = s_{B'} C_{A,1} + q_{B'} G, \quad F_{BA,2} = s_{B'} C_{A,2} + (q_{B'} - s_{B'} loc_{B'}) pk_A. \tag{10}$$

Next, the actual and future locations are converted to a single string and encrypted using AES, as in PPID (5). The UA appends this encrypted string to the response. Finally, the response message is digitally signed with the same ephemeral private key used for shared key generation. The response message is then broadcast by the UA.

In the verification and violation detection phase, the CI receives the broadcast responses from UAs. The CI verifies the authenticity, integrity, and validity of the message as in PPID. Then, it decrypts the encrypted current and future locations to perform equality testing

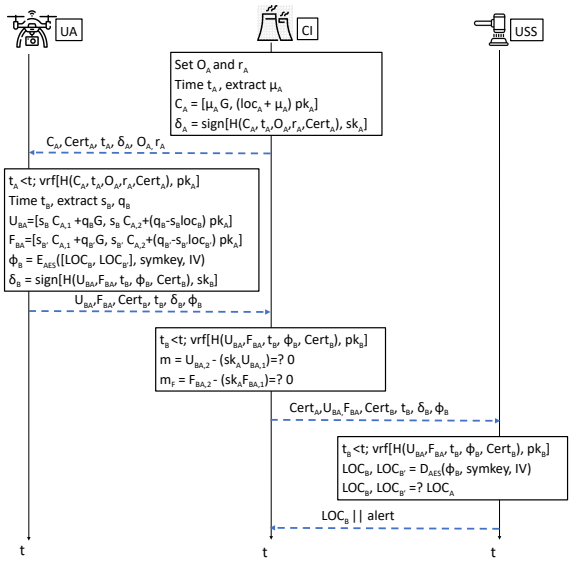


Figure 3: Sequence diagram of the e-PPID protocol.

and obtain $m = loc_A - loc_B$ and $m_F = loc_A - loc_{B'}$. Similar to the PPID procedure, CI follows the following steps:

(1) The CI first checks if m is an infinity point of the elliptic curve through Eq. 8, i.e., if the UA is invading the no-fly area. If so, it skips the next steps and sends a request to the USS to get the actual location of the UA.
(2) If $m \neq O$, then it is not an infinity point and the UA is not invading. The CI further checks if m_F is an infinity point, to infer whether the UA may invade in the future. If $m_F = O$, it then requests the USS to verify this event and generate an alert certificate for the drone.
(3) If neither m nor m_F are infinity points, it means that the UA is neither invading now nor it will invade in the future. The CI then terminates the protocol.

One of the main advantages of e-PPID is that the mapped future location and its corresponding computation are only done if the UA moves out of the circle, i.e., if the current and future mapped locations are different (also when the UA gets into a no-mapping zone). If the UA stays in the same circle, then the e-PPID protocol follows PPID, i.e., not generating and including F_{BA} in the encrypted location report.

Drawbacks. With e-PPID, the CI can be aware of a possible future violation, having an opportunity to prepare for it in advance. However, the additional information increases the computation and communication costs. Most importantly, the predicted location is based on the maximum speed and current angle, which, in case of remotely-piloted UAs, might change in the near future. For instance, the UA might turn to change its course, but the response message will not reflect it. Thus, the CI will detect a violation that might not occur, incurring additional communication overhead.

4 EVALUATION

In this section, we evaluate PPID and e-PPID. We provide a security analysis of the protocols in Section 4.1. We then describe our testbed and environmental setup in Section 4.2. We then show the performance of the protocols in Section 4.3, analyzing the run time, communication

cost, and memory requirements on a resource-constrained device. Notice that no other solution in the literature is readily available for application in our considered scenario. Therefore, we cannot include the comparison with other state-of-the-art solutions.

4.1 Security Analysis

Previous work already proved formally the security of the main building block of our proposal, i.e., [22], via mathematical proofs. However, in principle, the combination of such building blocks with other cryptography primitives used in our work (e.g., encrypted location reports) might jeopardize the overall security of the proposed scheme. When combining secure cryptography protocols into new ones, formal logic verification tools such as *ProVerif* allow the identification of possible vulnerabilities, making such tools the preferred solution over redundant mathematical discussions. Such a choice also aligns with other works in the very recent literature [25], [26].

Overall, *ProVerif* builds on two main assumptions: (i) the atomic cryptography primitives adopted in the protocol are secure, and (ii) the attacker has full access to the algorithms and the public values used in the protocol and to the communication link, where it can read all messages and inject its own ones. Based on such considerations, *ProVerif* applies automated procedures to find vulnerabilities in the logical usage of secure cryptography primitives. When a vulnerability is found, the tool also provides a step-by-step description of the attack.

In our case, we implemented our solution in ProVerif by modelling all the three entities described in Sec. 3.4, and we tested the confidentiality of the location loc_B of the drone during the execution of the protocol. Recall that *ProVerif* provides the output *not attacker(elem[]) is true* when the attacker is not in possession of the value of *elem*, while it provides the output *not attacker(elem[]) is false* when the attacker can obtain the value of *elem*. Moreover, *ProVerif* provides the output *weak secret(elem[]) is true* when the attacker cannot launch offline guessing attacks on the value *elem*, and vice-versa, it provides the output *weak secret(elem[]) is false* when offline guessing attacks on the value *elem* are possible.

Fig. 4 shows the output provided by ProVerif when testing the events described above. *ProVerif* verifies that: (i) the location of the drone, namely, loc_B, is not exposed to the attacker; and (ii) the way loc_B is used in the proposed protocol protects against offline guessing attacks, thus confirming our claimed security properties. We release the code of the security verification of the protocol in *ProVerif* as open source at https://github.com/ssciancalepore/ppid, allowing interested readers to replicate and verify our findings.

```
Verification summary:
Weak secret locB is true.
Query not attacker(locB[]) is true.
```

Figure 4: Excerpt of the output provided by the *ProVerif* tool.

4.2 Environmental Setup

We implement our two proposed protocols, PPID and e-PPID, in an actual proof-of-concept, using OpenSSL as the framework for Elliptic Curve Cryptography (ECC) in C. Our evaluation focuses on runtime performance, communication cost, and memory usage. We consider some key factors such as the security level of protocol, variety of key sizes, and number of iterations. We assume that the CI and USS have largely available resources, and thus, investigating the overhead of the protocol on their side is not interesting. We consider a scenario where, for e-PPID, location prediction accounts for a future time span of one second, i.e., $\Delta_t = 1$s. On the one hand, e-PPID can support any future time span. On the other hand, such a choice allows us to limit the prediction error when unexpected events causing a modification of the drone trajectory occur.

Environment To evaluate the overhead that the protocols have on an actual constrained device, we run our experiments on an embedded device. We use a Raspberry Pi 3 model B Rev 1.2, which is constrained in terms of memory and computation power. In particular, it is equipped with 4 processors running at 1.2 GHz, 1 GB of RAM, and 8 GB of SD card storage. This choice aligns with the current literature [27] and allows us to understand the efficiency and effectiveness of our protocols in constrained environments and to optimize them by choosing adequate security configurations.

Security Level As our protocol is based on ECC, we examined the effect of various key sizes on the performance of the protocol, i.e., 128,160, 192, 256, and 384 bits. The lower key sizes, i.e., 128 and 160 bits, might not provide adequate security; however, we decided to test them so as to compare the related performance with the ones achievable through larger key sizes. Similarly, the largest key size of 384 bits often provides a too high level of security, but it has certain drawbacks, possibly significantly affecting the runtime performance of our solutions. Hence, such an evaluation allowed us to balance security and performance. On top of that, AES-128 is also used for sharing the actual (and future) location of the drone with the USS. We also measured the performance of the main atomic cryptographic operations involved in our protocols, i.e., ECC Addition and Multiplication.

Precomputations It is interesting to note that some of the operations required by the proposed protocols can be precomputed, i.e., they can be executed offline, and their result can be stored in the local memory of the device(s). Such a strategy trades off storage with performance, boosting the computation time at the expense of larger memory overhead. Overall, it is possible to pre-compute all the values which do not depend on the runtime parameters of the drone. For instance, in the verification and response generation phase of PPID and e-PPID, where the drone generates the response to the challenge, the operation $q_B G$ (and $q_{B'} G$ for e-PPID) stays the same and does not depend on any dynamic protocol value, as G is the parameter of the curve and q_B (and $q_{B'}$) is a nonce. Generating nonces and doing multiplication during protocol execution requires high computation time, as multiplication operations on elliptic curves are more expensive than additions. Figure 5 confirms such an intuition, by showing the execution time of additions and multiplications over ECC on our resource-constrained device, with various elliptic curves. Although such results might appear redundant, they help show the overhead that we can get when executing such operations at deployment time, supporting our pre-computation strategy.

Therefore, before deployment, the drone can generate and save a set of nonces and their multiplication through the curve parameter. This operation increases the memory overhead of the protocol on the drone, but it reduces the corresponding execution time of the protocol. As the duration of a mission is typically known, the administrator of

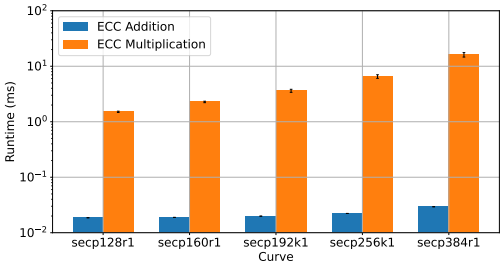


Figure 5: Runtime of ECC additions and multiplications, with various curves.

the drone can identify how much pre-computed values to store not to sacrifice security. Also, recall that equipping a drone with larger storage space can be done flexibly, by simply providing a larger SD card, without significantly affecting its weight and lifetime.

4.3 Results

In this section, we show the runtime performances of our protocols.

4.3.1 Computation Cost. We implement the following tests to evaluate our protocols.

- **Randomised Coordinates.** We generated 10,000 random coordinates around the location of the CI. Due to the random selection, the coordinates do not represent a real drone path. However, we use them to verify if the protocol is able to successfully detect violations when any of these points lie in the no-fly area. These tests are executed only through the PPID protocol, since there are no future coordinates to predict for e-PPID.

  Some errors are introduced in the results due to mathematical operations in the location encoding. For the overlapping circles, the drone selects the circle whose center is closest to its location. Our aim is to detect if any point lies inside the circle of CI. So, we can neglect the *no-mapping zone* as it is a point outside of the circle.
- **Fixed Coordinates.** To evaluate pre-computations's impact, we ran the protocol 10,000 times with fixed coordinates. As we are computing the runtime performance, we have chosen the coordinates such that every phase of PPID and e-PPID protocol is executed. Figure 6 compares the performance of the protocols with and without pre-computations, i.e., how much time the CI needs to detect (possible) invasions of the NFZ with different protocol configurations. We see that precomputations reduce the runtime of the protocols. The bar graph shows the average values over 10,000 iterations, with vertical lines representing the 95% confidence interval of the measurements.
- **Real Drone Flight Dataset.** We have tested the correctness of the e-PPID protocol also using real data, i.e., the ICMCIS dataset of OTAN, obtained from a real flying drone [28]. The data were originally provided in the context of a challenge, where the participants' task was to track, classify, and identify Class I UAs as they flew within a defined area. The available data include the log files of the UAs providing, among the others, information about the specific location (latitude, longitude, and altitude) of the UA at a given time (reported through a timestamp with a precision of 1 μs), and the instantaneous readings of the speed of the UAV (along the three-axis x-y-z). Such information is available with an average frequency of 90 msec. We chose a random location for the CI and ran the protocol to detect a violation, if any. Figure 7 shows the safe (i.e., non-invading) flight path of the drone in green, with the protocol detecting the future and current violation, in yellow and red, respectively. This scenario allowed us to understand the behavior of the protocol and its capabilities of correctly predicting future violations. It is worth noting that a violation of the current location has a higher precedence than a future violation, meaning that when a drone is invading, we do not care about the future violation. Hence, it is understandable that when a drone leaves the no-fly area both current and future locations will be marked as safe. This holds unless the drone moves again toward the no-fly area.

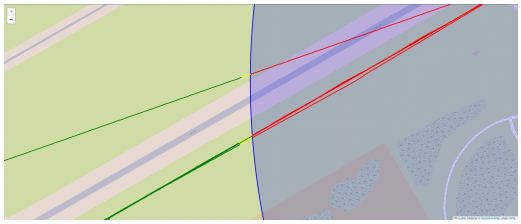
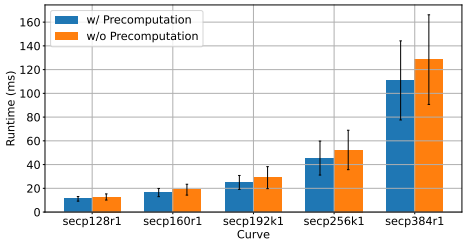
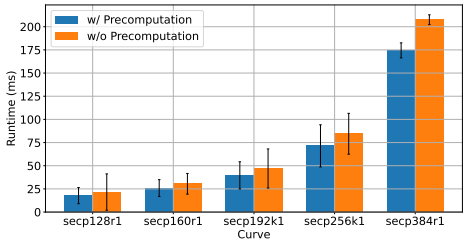


Figure 7: Violation detection of a drone during its flight. The path is green (no violations) when outside of the circle, yellow when close to the circle (future violation), and red (violation detected) when inside the circle.



(a) Performance of the PPID protocol.



(b) Performance of the e-PPID protocol.

Figure 6: Execution time of the runtime phase of the protocol, with various ECC curves.

4.3.2 Communication Cost. The communication overhead plays an important role in the performance because the drone might have to respond to several CIs simultaneously. Thus, higher communication overhead would reduce the available bandwidth and hence cause delays that may impact drone operations negatively. Figure 8 shows the communication overhead of our protocols for different key sizes. We see that the e-PPID protocol requires higher communication overhead compared to PPID, as it requires additional messages to be delivered and a slightly bigger message size, due to AES encryption of the future location.

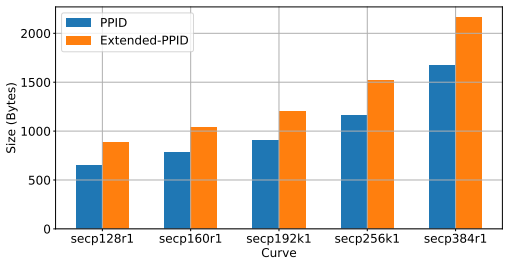


Figure 8: Total communication overhead (TX + RX) on the drone for both protocols.

4.3.3 Memory Usage. In our context, the memory usage of the protocols refers to the amount of RAM used during the execution of the protocol. Figure 9 compares the amount of memory used by the proposed two protocols. Each figure compares the amount of memory required for the execution of the complete protocol and the memory overhead of the phases of the protocols executed on the drone. The difference between the two protocols is hardly noticeable. Indeed, the OpenSSL library is the component that requires most of the memory, while the protocols require very small additional memory, Also, the e-PPID protocol requires slightly more memory due to the additional messages involved in the protocol compared to PPID.

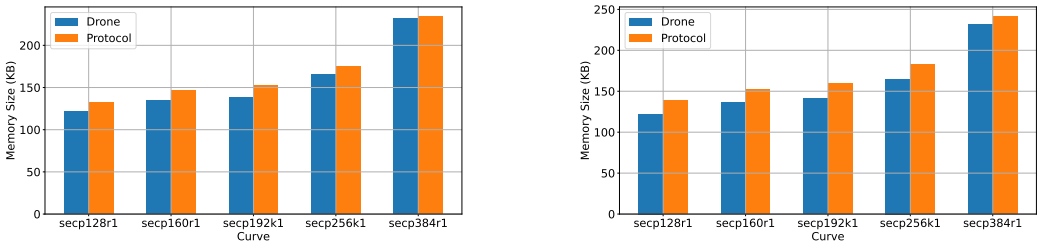
5 RELATED WORK

Drone Location Privacy. Svaigen et al. considered the location privacy issue in the Internet of drones [29]. However, their solution mostly focuses on the anonymity of drones rather than location privacy. In [30], the authors leverage a topology-based dummy generator to plan the drone trajectory while preserving its privacy. The solution is based on k-anonymity and cannot be used to detect proximity between a drone and a CI. Another work aimed at solving the location privacy issues of drones is [31], where the authors consider the problem of location privacy where the drone collaborates with a ground vehicle to save energy while travelling. Also in this work, the authors consider k-anonymity as a solution, thus focusing on the anonymity of the drone rather than the possibility of detecting collisions. To the best of our knowledge, the only similar work in the literature that tackles the location privacy issue of drone and CI co-detection is [12]. The authors proposed using geo-indistinguishability as a statistical approach to allow a CI to detect the presence of drones in a no-fly area. However, as previously discussed, a statistical approach suffers from the presence of a significant number of false positives, thus creating remarkable management overhead by the CI.

Other works in the literature considered the problem of privacy-preserving collision avoidance for vehicular networks [32, 33] and co-presence verification [34]. However, we notice that drones move in three-dimensional space, rather than in two-dimensional space as road vehicles do. Therefore, these solutions are not readily implementable in our considered scenario.

Privacy Preserving Collision Avoidance. Our work is also partly inspired by the recent work in [20] on collision avoidance for autonomous UAVs. Compared to the cited work, we provide several modifications and improvements to make such a solution fit to our problem and context. First, we do not deal with collision avoidance, but with privacy-aware intrusion detection. Second, we deal with any kind of UAs and not only autonomous ones; thus, we use circles rather than capsules, allowing CIs to specify the desired size of the no-fly area. Finally, we include also *Encrypted Location Reports* in the protocol design, so as to allow the USS to obtain the current position of the invading UAV, if needed. Conversely, the straightforward application of the proposal in [20] to the problem tackled in this manuscript would not allow any entity to disclose the location of the invading drone, being useless for applying the following countermeasures. Similar problems have been also considered for satellite collision estimation, e.g., in [35], but those solutions are typically very computationally expensive when used on constrained drones.

Private Proximity Testing. The protocols proposed in this contribution integrate as a main building block the private proximity testing solution proposed by the authors in [22]. On the one hand, we notice that the straightforward application of such a primitive in this context would not allow us to address our problem, as the cited solution is



(a) PPID Protocol. **(b) e-PPID Protocol.**

Figure 9: Memory overhead required by the protocols, with various ECC curve sizes.

intended for location-based services. On the other hand, we notice that many solutions are available in the literature for private proximity testing, e.g., [36] and [37], to name a few. However, PPID builds on top of the solution in [22] as it is the only one which can be adapted for a broadcast communication scenario. All the others, instead, require multiple communication rounds, not being usable in our context.

6 CONCLUSION AND FUTURE WORK

This paper presents PPID, a protocol enabling Critical Infrastructure (CI) operators to detect drones unintentionally entering no-fly zones while preserving location privacy for both drones and CI operators. We also introduce e-PPID, a variant that predicts future violations with the same privacy guarantees. The protocols were verified for logical security using *ProVerif* and implemented as a proof-of-concept on a constrained device. Performance evaluation shows PPID achieves private violation detection in just 52.31 msec on the $secp256k1$ curve, making it secure and feasible for commercial drones. Future work will focus on protocol optimizations and detecting malicious drones using forged location data.

ACKNOWLEDGMENTS

Project funded by the European Union under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 341 of March 15, 2022 of Italian Ministry of University and Research – NextGenerationEU; Code PE00000014, Concession Decree No. 1556 of October 11, 2022 CUP D43C22003050001, Project "SEcurity and RIghts in the CyberSpace (SERICS) - Spoke 7 Infrastructure Security - Visible Light Communication for Secure Vehicle-to-Everything Communication - VisiCar" – Beneficiary's CUP: C99J24000250008. Also supported by the European Union's Horizon Research and Innovation Programme, as part of the Project LAZARUS under Grant 101070303

REFERENCES

[1] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 881–896.

[2] S.-G. Kim, E. Lee, I.-P. Hong, and J.-G. Yook, "Review of intentional electromagnetic interference on uav sensor modules and experimental study," *Sensors*, vol. 22, no. 6, p. 2384, 2022.

[3] M. A. El-Zawawy, A. Brighente, and M. Conti, "Authenticating drone-assisted internet of vehicles using elliptic curve cryptography and blockchain," *IEEE Transactions on Network and Service Management*, 2022.

[4] ——, "Setcap: Service-based energy-efficient temporal credential authentication protocol for internet of drones," *Computer Networks*, vol. 206, p. 108804, 2022.

[5] H. Li, G. Johnson, M. Jennings, and Y. Dong, "Drone profiling through wireless fingerprinting," in *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2017, pp. 858–863.

[6] O. A. Ibrahim, S. Sciancalepore, and R. Di Pietro, "Noise2Weight: On detecting payload weight from drones acoustic emissions," *Future Generation Computer Systems*, vol. 134, pp. 319–333, 2022.

[7] BBC, "Gatwick Airport: Drone grounds flights," https://www.bbc.com/news/uk-england-sussex-46623754, 2019, accessed: 2024-Sep-11.

[8] Reuters, "Armed drone fired at Abha airport," https://www.reuters.com/world/middle-east/saudi-led-coalition-says-intercepts-armed-drone-fired-abha-airport-2021-05-10/, 2021, accessed: 2024-Sep-11.

[9] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici, "SoK: Security and privacy in the age of commercial drones," in *IEEE Symposium on Security and Privacy*. IEEE, 2021, pp. 1434–1451.

[10] FAA, "Remote identification of drones," https://www.faa.gov/uas/getting_started/remote_id, 2021, accessed: 2024-Sep-11.

[11] US Court of Appeals, "Tyler Brennan v. Stephen Dickson, No. 21-1087 (D.C. Cir. 2022)," https://law.justia.com/cases/federal/appellate-courts/cadc/21-1087/21-1087-2022-07-29.html, 2022, accessed: 2024-Sep-11.

[12] A. Brighente, M. Conti, and S. Sciancalepore, "Hide and Seek: Privacy-Preserving and FAA-compliant Drones Location Tracing," in *Proc. of International Conference on Availability, Reliability and Security*, 2022, pp. 1–11.

[13] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "ARID: Anonymous Remote IDentification of Unmanned Aerial Vehicles," in *Annual Computer Security Applications Conference*, ser. ACSAC '21, 2021, p. 207–218.

[14] AIN, "RemoteID Drone Rule Raises Privacy Concerns," https://www.ainonline.com/aviation-news/business-aviation/2021-01-22/nbaa-remote-id-drone-rule-raises-privacy-concerns, 2021, accessed: 2024-Sep-11.

[15] Chang, X. et al., "A surveillance system for drone localization and tracking using acoustic arrays," in *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2018, pp. 573–577.

[16] A. Hussain, N. Abughanam, S. Sciancalepore, E. Yaacoub, and A. Mohamed, "Jammer Localization in the Internet of Vehicles: Scenarios, Experiments, and Evaluation," in *Proceedings of the 12th International Conference on the Internet of Things*, 2022, pp. 73–80.

[17] S. Oh and M. Gruteser, "Multi-node coordinated jamming for location privacy protection," in *2011-MILCOM 2011 Military Communications Conference*. IEEE, 2011, pp. 1243–1249.

[18] S. Oh, T. Vu, M. Gruteser, and S. Banerjee, "Phantom: Physical layer cooperation for location privacy protection," in *Proc. IEEE INFOCOM*. IEEE, 2012, pp. 3061–3065.

[19] M. Lin, W. Wang, and C. Liu, "Preventing hostile toa/tdoa localization with af relay," *IEEE Communications Letters*, vol. 27, no. 4, pp. 1085–1089, 2023.

[20] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "PPCA-Privacy-Preserving Collision Avoidance for Autonomous Unmanned Aerial Vehicles," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[21] M. Lisi, "Some remarks on the cantor pairing function," *Le Matematiche*, vol. 62, no. 1, pp. 55–65, 2007.

[22] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh *et al.*, "Location Privacy via Private Proximity Testing," in *NDSS*, vol. 11, 2011.

[23] V. Gayoso Martínez, L. Hernández Encinas, and A. Queiruga Dios, "Security and practical considerations when implementing the elliptic curve integrated encryption scheme," *Cryptologia*, vol. 39, no. 3, pp. 244–269, 2015.

[24] R. Haakegaard and J. Lang, "The elliptic curve diffie-hellman (ecdh)," *Online at https://koclab. cs. ucsb. edu/teaching/ecc/project/2015Projects/Haakegaard+ Lang. pdf*, 2015.

[25] L. Hirschi and C. Cremers, "Improving automated symbolic analysis of ballot secrecy for e-voting protocols: A method based on sufficient conditions," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 635–650.

[26] E. Wisse, P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "A^2RID-Anonymous Direct Authentication and Remote Identification of Commercial Drones," *IEEE Internet of Things Journal*, 2023.

[27] D. R. George, S. Sciancalepore, and N. Zannone, "Privacy-Preserving Multi-Party Access Control for Third-Party UAV Services," in *Proc. of ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '23, 2023, p. 19–30.

[28] NATO, "Drone identification and tracking," https://www.kaggle.com/c/icmcis-drone-tracking/overview, 2021, accessed: 2024-Sep-11).

[29] A. R. Svaigen, A. Boukerche, L. B. Ruiz, and A. A. Loureiro, "Mixdrones: A mix zones-based location privacy protection mechanism for the internet of drones," in *Proceedings of the 24th International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2021, pp. 181–188.

[30] ——, "A topological dummy-based location privacy protection mechanism for the internet of drones," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1–6.

[31] A. R. Svaigen, A. Boukerche, L. B. Ruiz, and A. A. F. Loureiro, "Mixride: An energy-aware location privacy protection mechanism for the internet of drones," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 3527–3532.

[32] Z. Qin, Y. Li, X. Ye, J. Zhou, M. Cao, and D. Chen, "Ecas: an efficient and conditional privacy preserving collision warning system in fog-based vehicular ad hoc networks," *CCF Transactions on Networking*, vol. 3, pp. 205–217, 2020.

[33] L. Nkenyereye, C. H. Liu, and J. Song, "Towards secure and privacy preserving collision avoidance system in 5g fog based internet of vehicles," *Future Generation Computer Systems*, vol. 95, pp. 488–499, 2019.

[34] C. Vaas, M. Juuti, N. Asokan, and I. Martinovic, "Get in line: Ongoing co-presence verification of a vehicle formation based on driving trajectories," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 199–213.

[35] B. Hemenway, S. Lu, R. Ostrovsky, and W. Welser Iv, "High-precision Secure Computation of Satellite Collision Probabilities," in *Security and Cryptography for Networks: 10th International Conference, SCN 2016, Amalfi, Italy, August 31–September 2, 2016, Proceedings 10*. Springer, 2016, pp. 169–187.

[36] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography and Data Security*, 2010, pp. 143–159.

[37] P. Kotzanikolaou, C. Patsakis, E. Magkos, and M. Korakakis, "Lightweight private proximity testing for geospatial social networks," *Computer Communications*, vol. 73, pp. 263–270, 2016, online Social Networks.