University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Cybersecurity

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

Supervisor
Prof. Alessandro Brighente
University of Padova

Co-supervisor
Prof. Savio Sciancalepore
Eindhoven University of Technology

Master Candidate
Harshul Vaishnav

Student ID
2005556

Academic Year
2022-2023

"I believe in our digital era, privacy is a fundamental human right."
— Tim Cook

Abstract

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

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

Contents

Listing of figures

Listing of tables

Listing of acronyms

UA . . . . . Unmanned Ariel

IV . . . . . Initialization Vector

CI . . . . . Critical Infrastructure

USS . . . . . Unmanned Service Supplier

ECC . . . . . Elliptic Curve Cryptography

AES . . . . . Advanced Encryption Standard

FAA . . . . . Federal Aviation Administration

PPID . . . . . Privacy Preserving Invasion Detection

ECDH . . . . . Elliptic-curve Diffie–Hellman Scheme

ECIES . . . . . Elliptic Curve Integrated Encryption Scheme

e-PPID . . . . . Extended Privacy Preserving Invasion Detection

1 Introduction

The availability of UA (a.k.a. drones[1]) on a commercial scale, as well as the possibility to equip them with IT-based modules, paved the way to the investigation of cyber-security and privacy for such mobile devices. Indeed, drones can be possible victims of cybersecurity attacks or deliverers of security and safety-threatening actions.

For 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].

When drones are used for threatening actions, famous examples include the attack to the Gatwick airport [6] or the attacks carried out in the US and Saudi Arabia in 2021 [7]. 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 in-

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

formation. Therefore, it is fundamental to protect the airspace from unauthorized access by UA [8]. To this aim, sensitive targets such as CI should be equipped with drone detection technologies.

To support UA detection and identification, the FAA mandates the use of *Remote ID* [9], 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, thus undermining drones' privacy. Therefore, *Remote ID* fully sacrifices drones location privacy on the altar of being able to promptly detect and identify misbehaving drones.

Motivation. Protecting the location privacy of drones and providing means to CI to detect the presence of intruders should not be a trade-off. Indeed, it is fundamental for drone users, especially for the ones performing sensitive commercial and military operations, to be able to fly drones in public spaces without necessarily disclosing sensitive information that may lead to tracking and profiling. At the same time, CI need to be able to detect the presence of intruding drones, so as to protect their assets and to avoid safety issues (e.g., a drone dropping a bomb over a nuclear plant area). Therefore, CI should receive location information from UA 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 [10]. 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 invasion 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 PPID, the first solution to simultaneously protect UA' privacy and allow CI to detect invasions from UA. 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 invasion detection, without sacrificing the privacy guarantee of both UA and CI.

We first propose our protocol and describe the information that UA and the CI should exchange. Then, we propose extended-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 and compute their execution time and memory requirements for different key lengths. Our results show that our protocols can run on resource-constrained devices with an average runtime of tens of milliseconds thus representing a suitable solution for fast invasion detection. Furthermore, they can run with limited communication costs (kByte order) and low memory overhead (hundred kByte order), hence being suitable for implementations on a resource-constrained device such as a drone.

We summarize our contributions as follows.

- We propose PPID, a novel protocol that allows a CI to detect an invasion from a UA while preserving the location privacy of the UA. Our solution provides means for the CI to notify the UA pilot about the invasion to have the UA change its course.

- We propose an extended version of PPID, i.e., extended-PPID, that aims at preventing future invasions. 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 asses their security against possible attacks by using the *ProVerif* tool. 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 an invasion requiring approx. 52ms and 84ms in the worst-case scenario, respectively, for a 256 bits security level.

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

2

System And Threat model

We provide here a description of the entities and assumptions of our work. In particular, we describe the system model in Section 2.0.1 and the threat model in Section 2.0.2.

2.0.1 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 invasions of the monitored area, namely the *no-fly area*, by unauthorized UA, to prevent eavesdropping of sensitive information both in terms of audio and video recordings from suitably equipped UA. 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 assume that 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 challenge, we assume that the drone replies to the CI with an encrypted version of its location as we explain in Chapter 3. Also, we assume the drone features a GPS receiver, so as to be able to compute its own

actual location, in terms of latitude, longitude, and altitude.

Another assumption is that the drones (and their pilots) 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 it is not currently possible to provide means for non-malicious UA to protect their location privacy while avoiding invading a no-fly area. Thus, our proposal provides a solution to this problem.

To support the decision on the invasion by providing precise information on the drone's location, we assume that the CI relies on the USS, i.e., a trusted service able to disclose the true location of the UA. Once the CI detects a possible invasion, it forwards to the USS the data needed to generate proof of an invasion. The USS can then obtain the actual drone location, verify the occurrence of a possible invasion and, if this is actually happening, report it to the CI. In the protocol description in Chapter 3, we discuss possible actions that the CI might undertake upon confirming the invasion.

2.0.2 Threat Model

In this paper, we consider two threats, i.e., an unaware pilot inadvertently flying his drone close to a CI area, and an active attacker trying to track a drone, as described below.

Unaware pilot, $\mathcal{A}_1$. This entity is represented by a pilot flying a drone for amateur purposes. During its flight, unintentionally, the drone might get close to a CI's area and undergo the risk of being damaged due to possible CI's anti-drone technologies. We assume that $\mathcal{A}_1$ cannot tamper with the UA firmware, e.g., it cannot send bogus location information. The latter is a reasonable assumption, as our model does not consider the presence of pilots with malicious intents, but rather users adhering to the existing regulations. As it is impossible for the CI to distinguish between a drone with malicious intent and a drone belonging to an unaware pilot, once an invasion is detected, the CI takes countermeasures against the drone. Thus, we assume

that $\mathcal{A}_1$ will move away from the CI area once receiving a warning message.

Active 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 UA, 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" [9]. Lastly, a company may want to track UA 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 [11]. 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, they are 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 invasion detection. Feasible solutions to detect such misbehavior might be based on the time difference of arrival for location estimation [12]. 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 CI Location Privacy. We note that considering only the exchanged information, our protocol also avoids direct disclosure of the location of the CI, besides the one of the UA. However, opposite to UA, 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, it is indeed feasible for any user 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) [12], [13]. Therefore, our protocol cannot provide location privacy for the CI. Conversely, due to the dynamic nature of the UA and their op-

portunistic interaction through the protocol, such localization techniques hardly allow drone localization.

3
Protocol Description

In this section, we describe our proposed protocols. We define the location encoding process in Section 3.0.1 and describe PPID in a nutshell in Section 3.0.2. Sections 3.0.3, 3.0.4, and 3.0.5 describe the steps of PPID. We then present e-PPID in Section 3.0.6. For the readers' convenience, we report in Table 3.1 the main notation used throughout the paper, with the corresponding description.

3.0.1 Location Encoding

Similarly to [14], PPID is based on a tessellation logic aimed at detecting proximity between UA and CI. To this aim, we leverage Location Encoding to encode geographic coordinates into points over a tasseled space. In particular, given the circular tessellation shown in Figure 3.1, we map the actual location coordinates of the considered entity to the location of the closer circle center.

To perform the equality test, we need to compute the difference between two values. If such a difference is zero, then the two values are equal, while they are not equal if the difference is non zero. To carry out equality testing in presence of a this tasseled space, we need a common coordinate system, such that any point within the circle corresponds to a single coordinate value,

Table 3.1: Notation used throughout the paper.

Notation	Description
D_n	ID of n-th UA
CI_n	ID of n-th CI
sk_n	Private key of the n-th entity.
pk_n	Public key of the n-th entity.
$Cert_n$	Public key certificate of the n-th entity.
$symkey$	Shared secret key, for AES cryptosystem.
IV	Initialization Vector, for AES cryptosystem.
$H(\cdot)$	Hash function.
$E[message, sk]$	Digital signature generation of message, using secret key.
$D[signature, pk]$	Digital signature verification of signature, using public key.
$E_{AES}[message, symkey, IV]$	Symmetric encryption of message using AES cryptosystem with symkey and IV.
$D_{AES}[cipher, symkey, IV]$	Decryption of cipher using AES cryptosystem with symkey and IV.
$\mathcal{E}$	Elliptic curve.
p	Prime number, size of elliptic curve field.
z_1, z_2	Parameters of the elliptic curve $\mathcal{E}$.
Z	Cyclic group of the curve $\mathcal{E}$.
G	Generator of the elliptic curve $\mathcal{E}$.
n	Order of the elliptic curve $\mathcal{E}$.
LOC_n	Actual location of n-th entity.
$LOC_{n'}$	Future location of n-th entity.
loc_n	Mapped actual location of n-th entity.
$loc_{n'}$	Mapped future location of n-th entity.
τ	Maximum validity time of a message on the USS.
O_n	Origin of the no-fly area of the CI, chosen by n-th entity.
r_n	Radius of the no-fly area of the CI, centered at O_n.
t_n	Timestamp selected by the n-th entity.
V_{max}	Maximum Speed of the UA.
K	Number of bits for the nonce
μ_n	Nonce extracted in $\mathbb{Z}_p$ selected by the n-th CI.
s_n, q_n	Nonces extracted in $\mathbb{Z}_p$ selected by the n-th UA.
$s_{n'}, q_{n'}$	Nonces extracted in $\mathbb{Z}_p$ selected by the n-th UA, for mapped future location.
C_n	Encrypted challenge generated by n-th active observer.
U_{mn}	Encrypted Response generated by the m-th UA to the challenge of the n-th observer.
F_{mn}	Encrypted Response generated by the m-th UA to the challenge of the n-th observer for future location.
δ_n	Message signature generated by the n-th entity.
φ_n	Encrypted location report generated by the n-th UA.
m, m_F	Verification code for the UA computed by the observer.

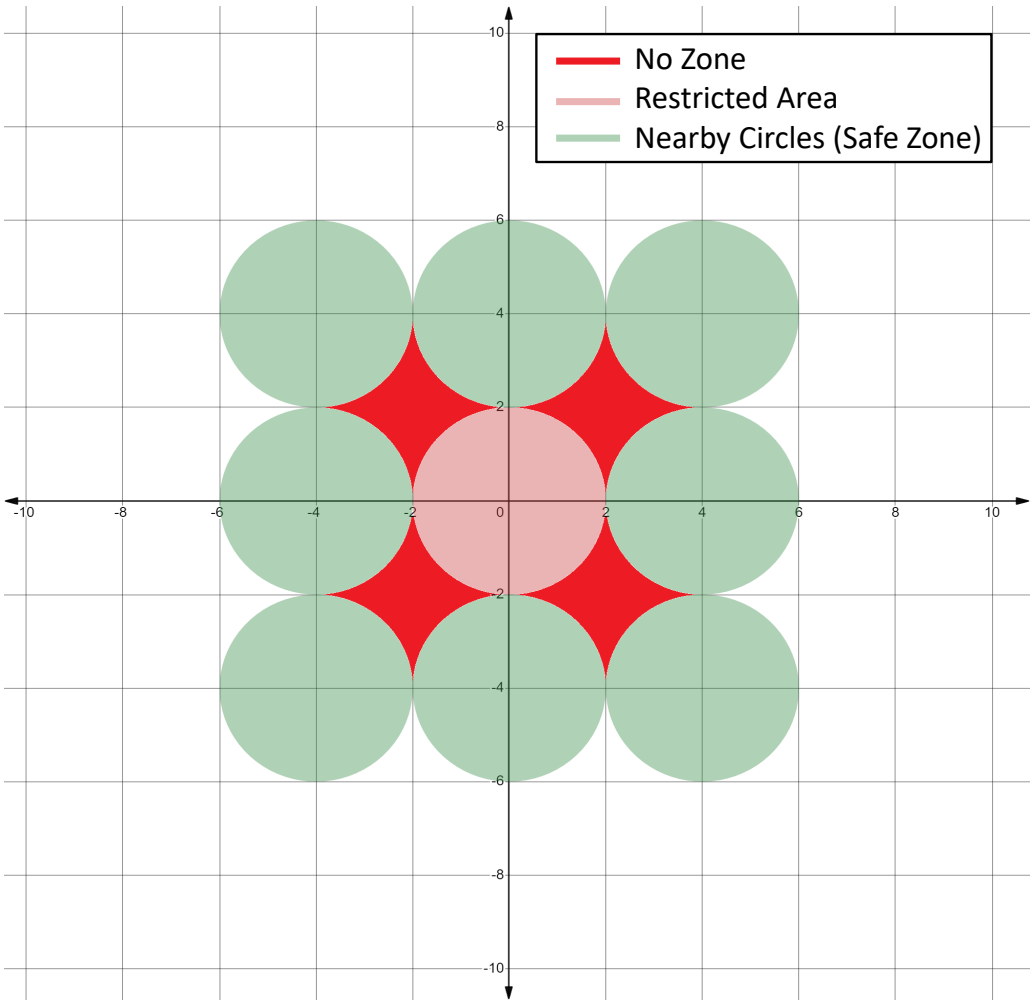


Figure 3.1: Example of location encoding, showing different zones, with the radius of 2 units for each circle.

i.e., the center of the circle. Such a strategy allows to uniquely identify a given area in this reference system. However, for running equality testing, we have to convert such coordinate into an integer, as cryptographic algorithms do not support floating point numbers. Such an integer value is termed as *mapped location*, and it is used across this paper.

We use Cantor pairing [15] to map the center coordinates to the *mapped location*, i.e., a scalar integer value. Given coordinates (x_n, y_n, z_n), the mapped location loc_n via Cantor pairing is

given by

$$loc_n = \frac{(f(x_n, y_n) + z_n)(f(x_n, y_n) + z_n + 1)}{2} + z_n, \tag{3.1}$$

where

$$f(\alpha, \beta) = \frac{(\alpha + \beta)(\alpha + \beta + 1)}{2} + \beta. \tag{3.2}$$

It is worth noting that Cantor Pairing works only for non-negative numbers. However, during the conversion of Geographic coordinates to a Cartesian system, we may get the negative coordinates which makes the pairing ineffective as

$$f(-\alpha, \alpha)\ and\ (-\alpha - 1, \alpha)$$

yield the same result. For the movement on Earth, it can be assumed that the chances of happening of such instances are very rare. A solution to this issue can be moving the system further towards the positive side such that there won't be any negative coordinate. This can be achieved because of the limited range of geographic coordinates.

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 zones[1], i.e., locations lying in no available circle (see red areas in Fig. 3.1). Recall that the aim of our solution is to detect co-location; thus, whenever the drone realizes that its actual location maps to a no-zone, it can be sure that there is no co-location.

It can also be noticed that the conversion of coordinates may generate some errors due to mathematical operations which may cause loss in precision. Especially, in the rounding operation during finding center of circle, the precision loss is higher. Due to this error, some part of the circle may overlap. This overlap can introduce confusion on which center to choose. Hence we choose the center which is closer to the entity, discussed further in Chapter 4.

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

3.0.2 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 and UA. Overall, we envision three phases, namely, the *Setup*, *Runtime*, and *Disclosure* phases.

In the *Setup Phase*, executed offline, both the CI and UA' operators set up the protocol by registering identification information with the USS, and receiving public parameters of the elliptic curve.

As detailed in Chap. 3.0.4, at run-time, 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 certificate, a timestamp, origin, and radius of the of no-fly area, and a signature (to ensure integrity). At message reception, any UA flying in the neighborhoods of the active observers, upon message integrity 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, an *invasion* is ongoing.

As detailed in Chap. 3.0.5, 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 invasion of the no-fly area of the reported entity. In case of invasion, 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.

3.0.3 Setup Phase

Figure 3.2 shows the steps executed by the involved entities during the setup phase. 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. Additionally, the UA receives L ephemeral public/private keys from the USS. 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.

3.0.4 Runtime Phase

Figure 3.3 shows the sequence diagram of the runtime and disclosure phase of the protocol, where the CI is entity A and UA is entity B. The protocol can be subdivided into 3 runtime sub-phases. In the following, we discuss each sub-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 for the validity of the challenge message. It also extracts a K bits nonce μ_A which is used for the encryption of CI's mapped location. The CI maps its actual location, to obtain loc_A[2], with the parameters provided to the USS, i.e., the origin, O_A and

[2]Note that, loc_A and LOC_A are not same. See table 3.1

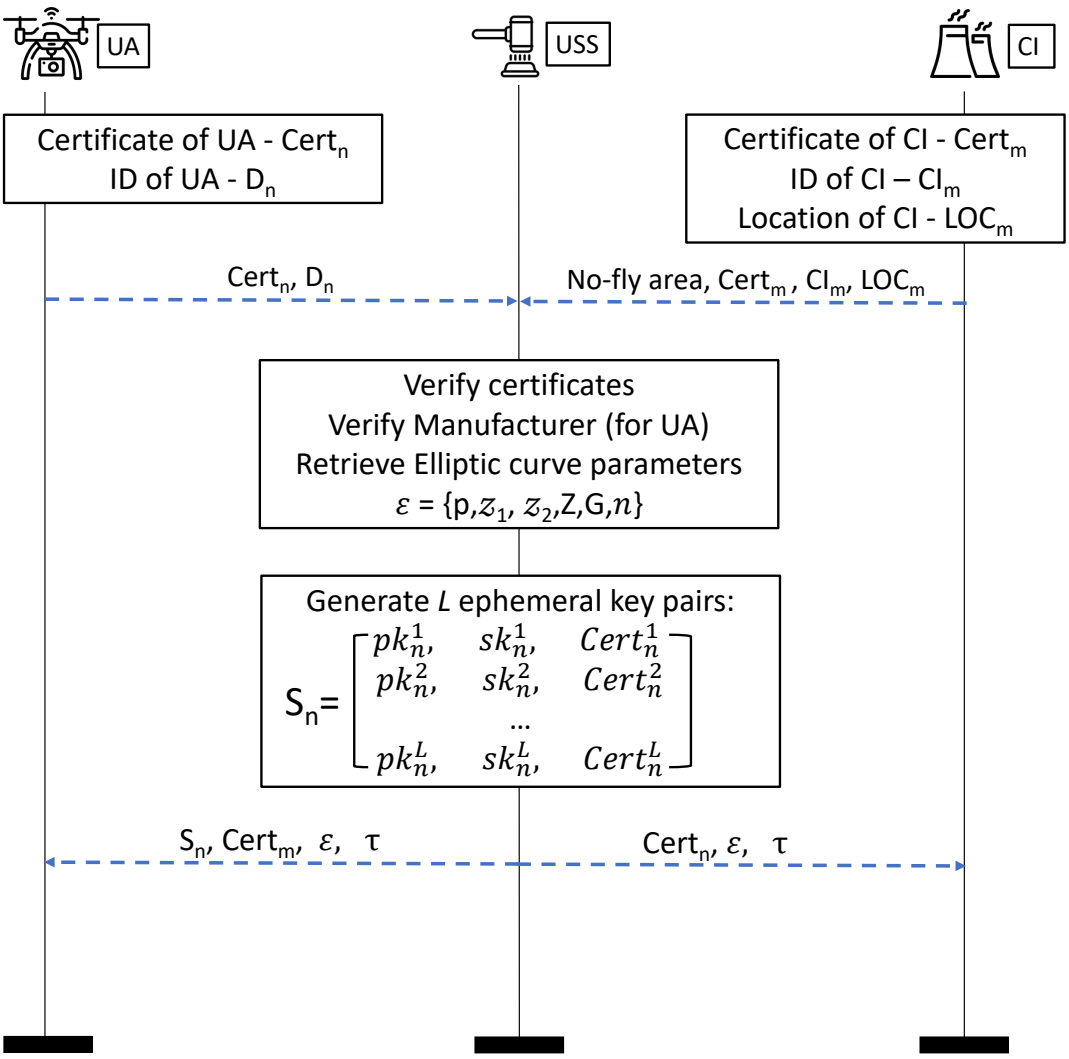


Figure 3.2: Sequence diagram of the setup phase of PPID.

radius, r_A of the circle obtained by the tessellation of Earth's surface, as described in Chapter 3.0.1. The mapped location is encrypted with CI's public key using El-Gamal elliptic curve cryptography as, in Eq. 3.3.

$$C_A = (C_{A,1}, C_{A,2}) = \left(\mu_A G, (loc_A + \mu_A)pk_A\right). \tag{3.3}$$

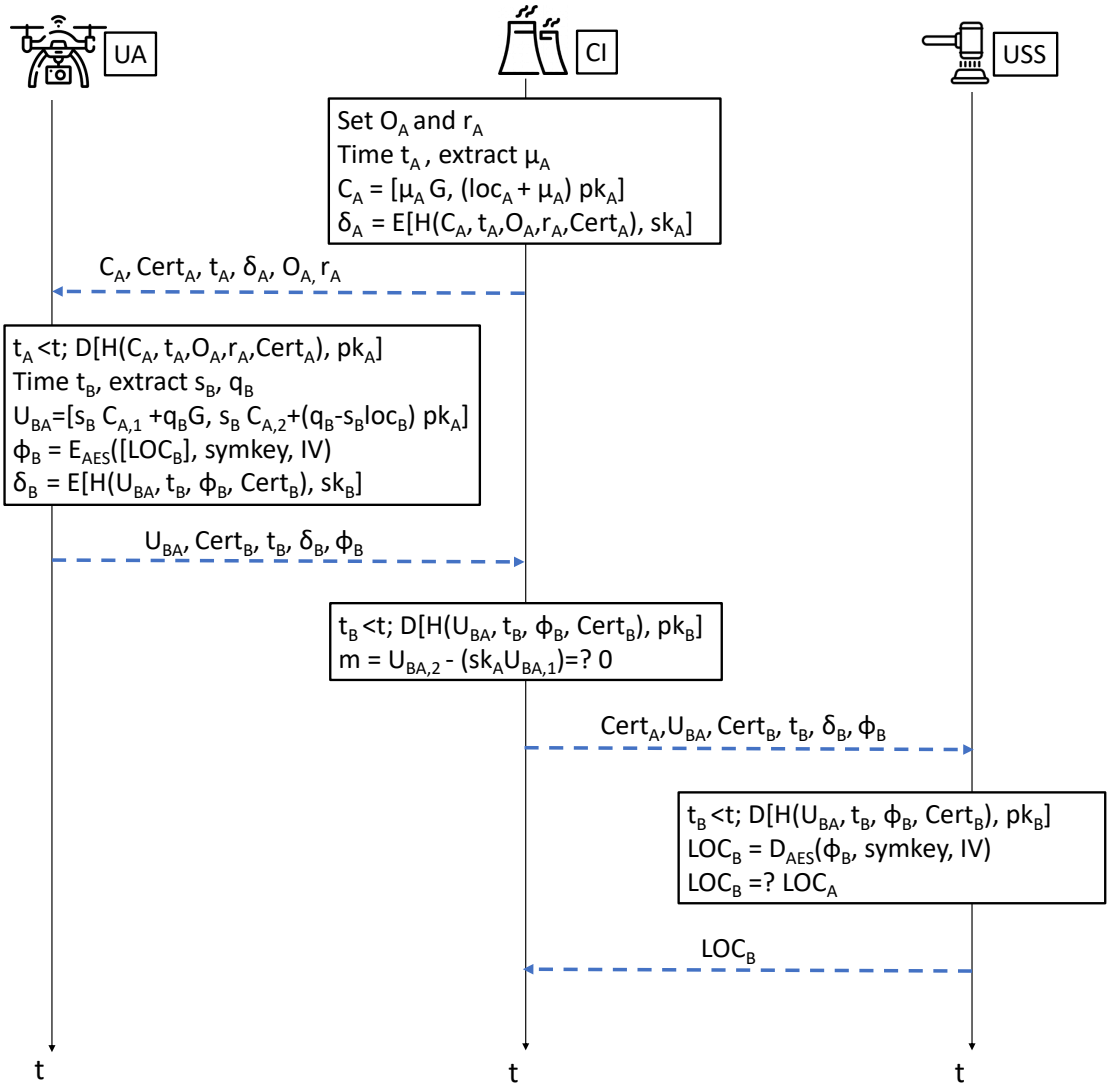


Figure 3.3: Sequence diagram of the PPID protocol.

The CI then signs the message using its private key, according to Eq. 3.4.

$$\delta_A = E\Big[H\Big(C_A, t_A, O_A, r_A, Cert_A\Big), sk_A\Big]. \tag{3.4}$$

Finally, the CI assembles and broadcast 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.

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 of the message using the public key of CI, via (3.5), along with the validity of the message.

$$D\big[\delta_A, pk_A\big] = H\big(C_A, t_A, O_A, r_A, Cert_A\big). \tag{3.5}$$

If the message is verified and received within the time limit, then the UA continues with the protocol; otherwise, it drops the message and stops the protocol execution.

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 calculates 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 (3.6).

$$\begin{aligned} U_{BA,1} &= s_B C_{A,1} + q_B G, \\ U_{BA,2} &= s_B C_{A,2} + (q_B - s_B loc_B) pk_A. \end{aligned} \tag{3.6}$$

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) [16]. ECIES is a hybrid encryption

scheme, meaning it uses two kinds of encryption in its process - symmetric and asymmetric. 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) [17]. This shared key is used to symmetrically encrypt the message. The other party repeats the same process to generate shared secret key and IV, i.e. ECDH, and symmetrically decrypt the message.

Any valid ECC key pair can be used for ECDH and hence it can be used in our protocol without generating any additional key. Also, we do not need to share the symmetric key, which makes it favorable for our protocol because the drone uses ephemeral keys and generates a new secret key every time. 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 (3.7) via the AES cryptosystem using the above-generated key and IV.

$$\varphi_B = E_{AES}([LOC_B], symkey, IV). \tag{3.7}$$

The value φ_B is appended to the response message. Finally, the response message is digitally signed in (3.8) with the same ephemeral private key used for the shared key generation. The response message is now broadcasted by the UA. It is worth noting that we don't require any additional key to maintain or share.

$$\delta_B = E\Big[H\Big(U_{BA}, t_B, \varphi_B, Cert_B\Big), sk_B\Big]. \tag{3.8}$$

Alternatively, to encrypt the actual location, the UA can generate a random secret key and then

encrypt this key with the public key of USS. For ECC, UA can choose a random point on curve and hash it with SHA-256 hashing algorithm. This results in 32 bytes string, of which first 16 bytes can be used for the key and the last 16 bytes for IV. With this method, an additional message, i.e. the encrypted symmetric key has to be sent along with the response.

Response Verification and Invasion Detection. Response messages such as the above one are broadcasted by all the UAs, who have received the beacon. These broadcast messages are received by the CI and processed to detect an invasion. 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 and reports it to the USS, otherwise, it continues the protocol to verify the integrity of the message using the public key of UA along with the validity of the message, as in Eq. (3.9).

$$D\big[\delta_B, pk_B\big] = H\big(U_{BA}, t_B, \varphi_B, Cert_B\big). \tag{3.9}$$

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 (3.10).

$$m = U_{BA,2} - (sk_A \cdot U_{BA,1}). \tag{3.10}$$

If $m = \mathcal{O}$, corresponding to the infinity point of the elliptic curve, then $loc_A = loc_B$ and the CI and UA lie in the same circle, leading to an invasion. If $m \neq \mathcal{O}$, then $loc_A \neq loc_B$, i.e., the CI and UA lie in two different circles and hence there is no invasion. As the invasion detection is performed in the encrypted domain, the CI cannot obtain any information on UA's location. If $m \neq \mathcal{O}$, 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.

3.0.5 Disclosure Phase

If the CI detects an invasion, it communicates it to the USS and forwards the response message received from the UA. Upon receiving the invasion request from the CI, the USS verifies the integrity of the message using the public key of the UA via Eq.(3.9), along with the validity of the message with the maximum validity time τ. If the message is verified and received within the time limit then 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. 3.11.

$$LOC_B = D_{AES}(\varphi_B, symkey, IV). \tag{3.11}$$

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 responds to the CI with 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 invasion message.

If the UA is invading, the CI can choose the action to undertake, i.e., either disarming the UA or alerting it. We envision the two following scenarios.

1. In the first scenario, 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 an invasion, the CI can decide on whether to disarm the drone or alert it. Such a situation can be visualised in Figure 4.3.

2. In the second scenario, the USS creates a small session between the invading UA and the CI. This allows UA to directly share its actual location with the CI which allows the CI to continuously track the UA within its territory. 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 invasion executed by the CI.

3.0.6 Extended-PPID protocol

In this section, we propose an extended version of PPID, namely, e-PPID, allowing to avoid future invasions 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 an invasion occurs, so as 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.

Remember that, based on PPID, the actual and future locations of the UA are only known by the UA and USS. The CI knows it only upon detecting an invasion. If both actual and future locations lie in the same non-invading circle, then the CI does not detect an invasion and will not communicate with the USS. Therefore, the CI does not have information related to the UA's location unless detecting an invasion on the present location. It is also worth noting that the time limit of the different parts of protocol execution needs to be increased as the additional messages in the protocol increase computation and require more time.

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 does not require any additional computation. The drone B first computes the future location

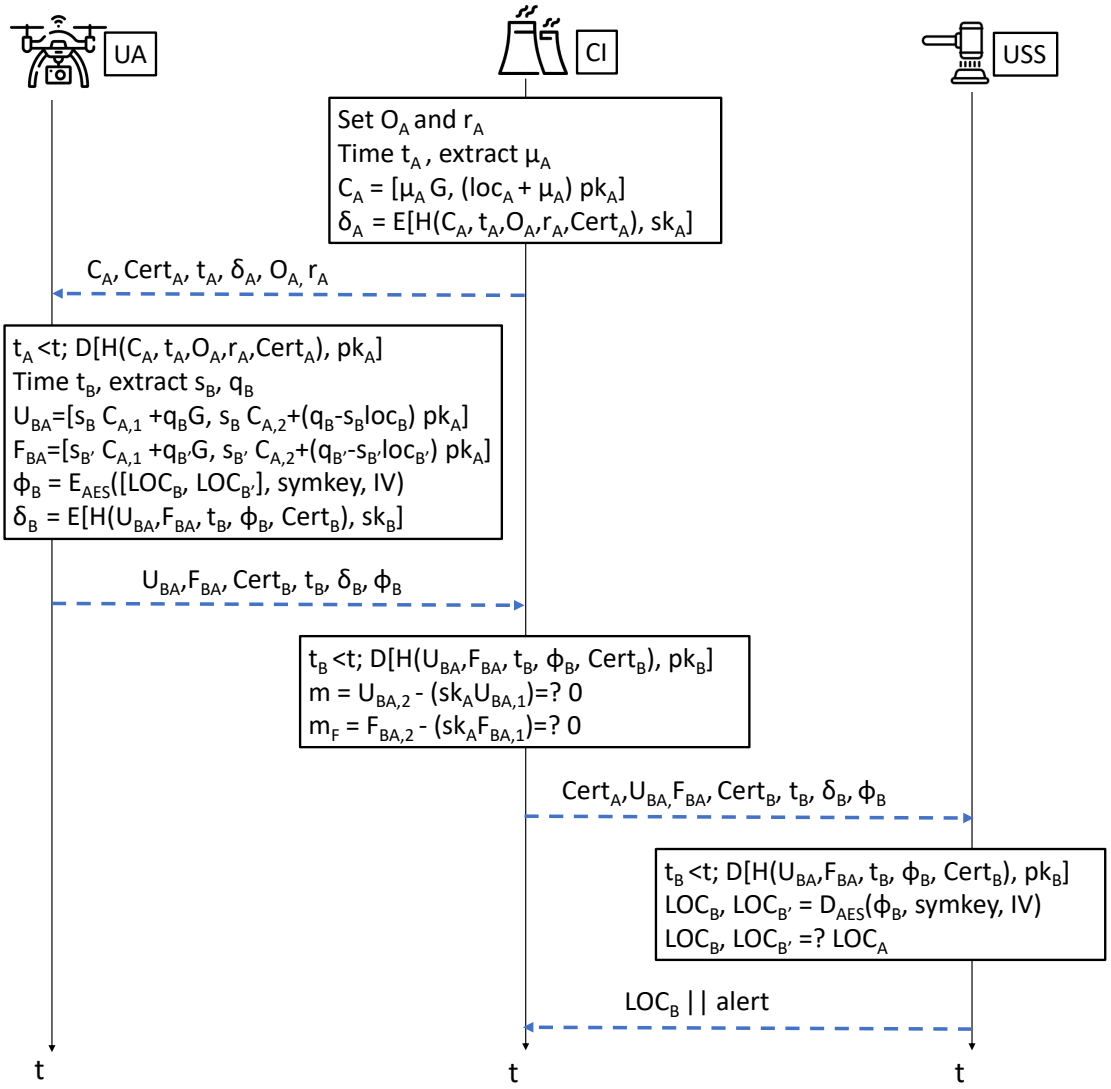


Figure 3.4: Sequence diagram of the e-PPID protocol.

$LOC_{B'}$. It then checks whether the mapped actual and future location lies in the same circle, a different circle, or in no 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 3.4 shows the steps of e-PPID. In the following, we describe the protocol phases of e-PPID. It 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. The computation is obtained as follows.

$$\begin{aligned} F_{BA,1} &= s_{B'} C_{A,1} + q_{B'} G, \\ F_{BA,2} &= s_{B'} C_{A,2} + (q_{B'} - s_{B'} loc_{B'}) pk_A. \end{aligned} \tag{3.12}$$

Next, the actual location and future location are converted to a single string and encrypted using AES, as in PPID (3.7). 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 broadcasted by the UA.

In the verification and invasion detection phase, the CI receives the broadcasted responses from UAs. The CI verifies the authenticity, integrity, and validity as in PPID. Then, it decrypts the encrypted responses containing both, mapped actual and future locations, to perform equality testing and obtain $m = (loc_A - loc_B)R$ and $m_F = (loc_A - loc_{B'})R$, where R is the result of remaining elements. 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. 3.10, 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 $m \neq \mathcal{O}$, 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 $m_F = \mathcal{O}$, it then sends a request to 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 actual and future mapped locations are different (also in case when the UA gets into a no zone). If the UA stays in the same circle, then 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 invasion, 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 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 an invasion 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 4.0.1. We then describe our testbed and environmental setup in Section 4.0.2. We then show the performance of the protocols in Section 4.0.3, analyzing the runtime, 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.

4.0.1 Security Analysis

We verify the security of the proposed protocol using the automated security verification tool *ProVerif*. Note that previous work already proved formally the security of some of the building blocks of our proposal, e.g., [18]. 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. Such a choice is

also in line with other works in the very recent literature [19], [20].

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 Chap. 3.0.4, 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. 4.1 shows the output provided by ProVerif when testing the events described above.

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

Figure 4.1: Excerpt of the output provided by the *ProVerif* tool.

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 [21], so as to allow interested readers to replicate and verify our findings.

4.0.2 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 programming language. Our evaluation focuses on runtime performance, communication cost, and memory usage. We take into account 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 hence the computation cost, communication cost and memory usage doesn't have any impact on them. We also assume that, for e-PPID, location prediction accounts for a future time span of one second, i.e., $\Delta_t = 1$s.

Environment. To simulate the impact that the protocols might have on an actual drone, we run our simulations 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. In line with the current literature [22], such a choice allows us to understand the efficiency and effectiveness of our protocols in constrained environments and to optimize them by choosing an adequate key size.

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, 128 and 160 bits might not provide adequate security; however, we decided to test them so as to compare the related performance with higher key sizes. Similarly, the higher key size of 384 bits would probably provide a too high level of security, but it has certain drawbacks, possibly significantly affecting the runtime performance. Hence, such an evaluation allowed us to find the balance between 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. For simulations, we have chosen the ECIES method for the sharing of actual location (and future location for e-PPID) as it is observed that the random secret key method takes approximately thrice the execution time of ECIES. We also measured the performance of cryptographic operations like 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 the protocol, 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 $q_{B'}$) are nonces. Generating nonces and doing multiplication during protocol execution requires high computation time, as multiplication operations on elliptic curves are more expensive than addition. Figure 4.2 confirms such an intuition, by showing the execution time of additions and multiplications over ECC on a resource-constrained device, with various elliptic curves.

Therefore, before deployment, the drone can generate and save a set of nonces along with 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 the drone can identify how much pre-computed values to store not to sacrifice security.

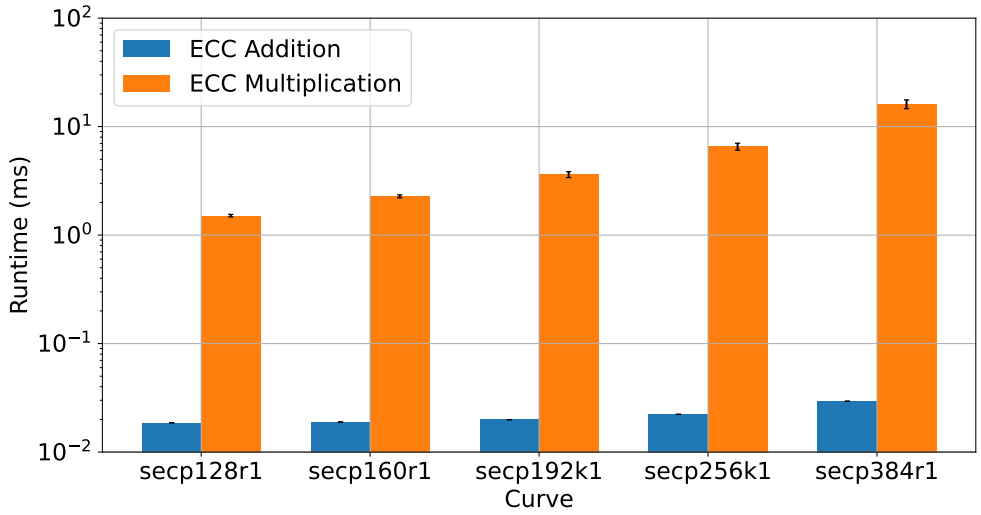


Figure 4.2: Runtime of ECC additions and multiplications, with various curves.

4.0.3 Results

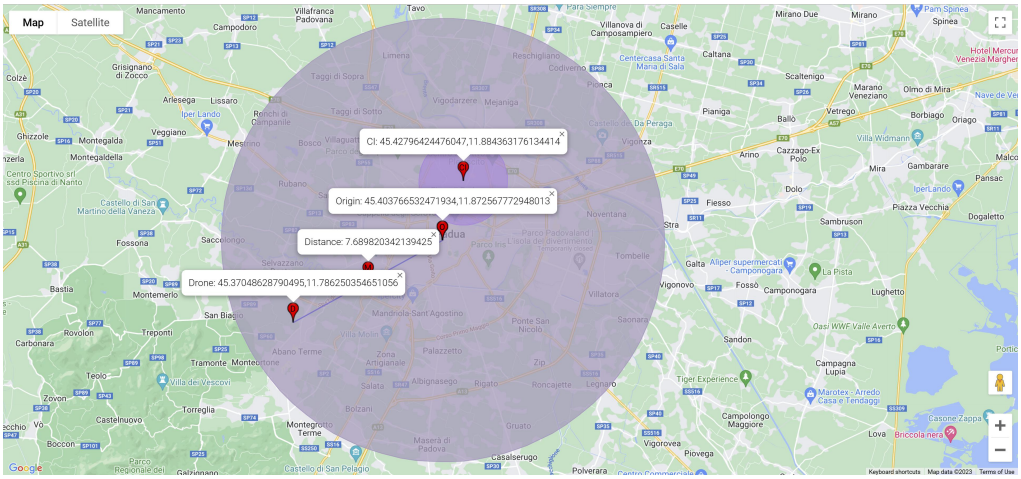
In this section, we show the runtime performances of our protocols. Results are primarily focused on the drone, as CI and USS do not have constrained resources. Also the results are for the worst-case runtime scenario.

Computation Cost

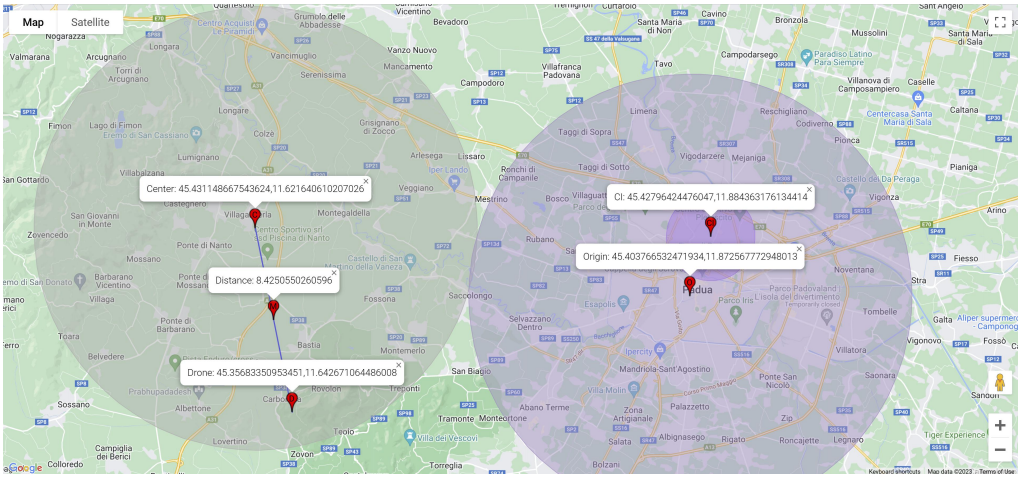
We implement the following tests to evaluate our protocols.

- **Randomised Coordinates.** We generated 10, 000 random coordinates around 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 an invasion 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.

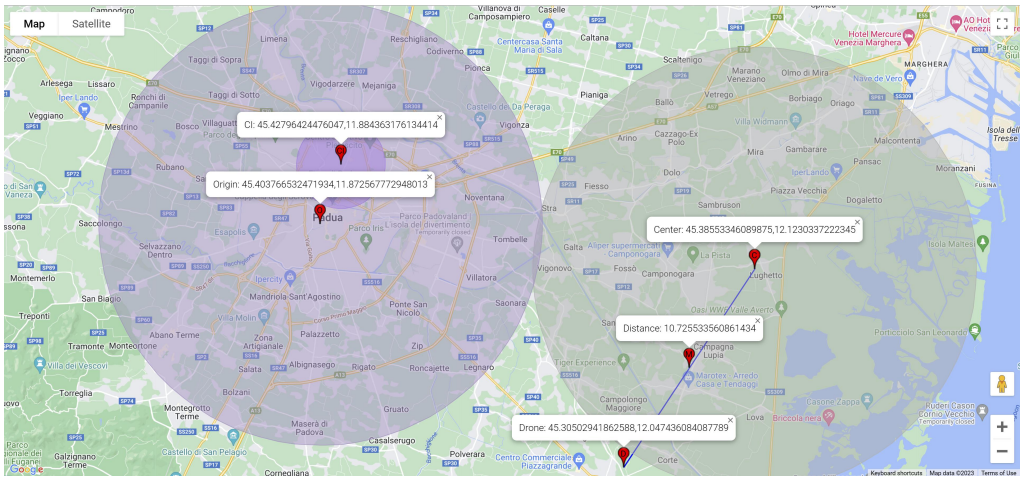
  Some errors are introduced in the results due to mathematical operations in location encoding. These errors can be visualized in figure 4.3, explained below. For the overlapping circles, the drone selects the circle whose center is closest to its location. Our aim is to detect if any point lies inside the circle of CI. So, we can neglect the *no zone* as it is a



(a) Drone is Invading the no-fly area



(b) Drone lies outside of no-fly area



(c) Drone lies in no zone

Figure 4.3: Visual representation of possible situations when comparing the location of a drone and a CI.

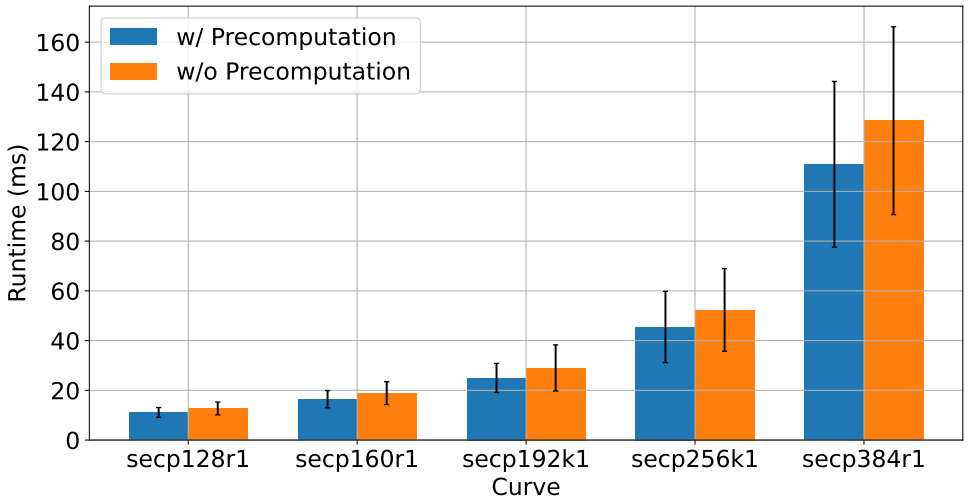
point outside of the circle.

Figure 4.3 shows the various situations that can occur in our simulations. The marker CI represents the location of the CI, marker O represents the Origin (i.e., the origin of the system providing the circular tessellation), and marker D represents the location of the drone. Given the center of the circle in which the drone lies into, marker M represents the midpoint of the distance between the circle's center and the drone, with its value being their distance.

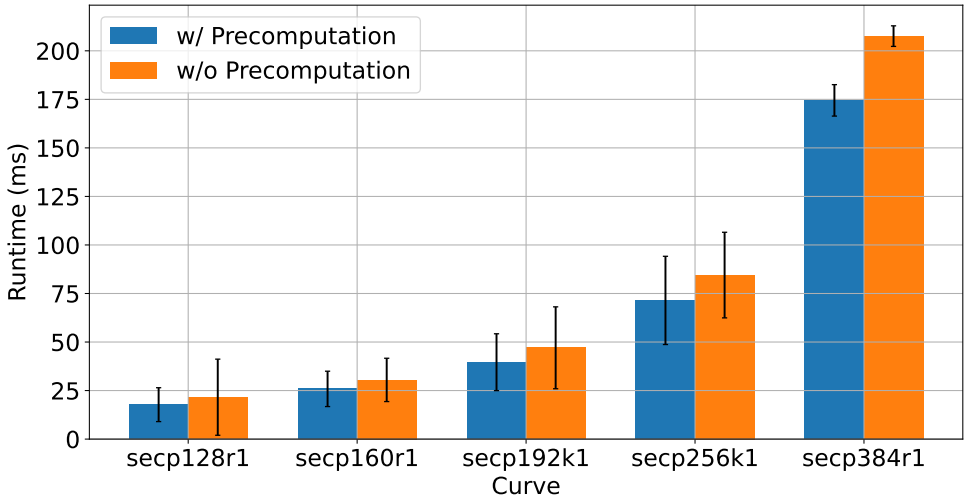
In the figure, the smaller circle around the CI shows the boundary of CI. It is the critical area of the CI, to defend against invasion. The no-fly area represents the entire circle. However, it may be different from the boundary of the CI. This allows CI to alert the drone. There are three possibilities for the location of the drone. One possibility is that the drone is invading the no-fly area, as shown in Fig. 4.3a. Another situation is when the drone lies in another circle, meaning the drone is not invading, as shown in Fig. 4.3b. Finally, the last one is when the drone is in a *no zone*, meaning it does not lie in any circle, as shown in Fig. 4.3c.

- **Fixed Coordinates.** To evaluate the positive impact of pre-computations, 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, thus obtaining the worst-case execution time of the protocols. Figure 4.4 compares the performance of the protocols with and without precomputations. 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 using the IMCIS dataset of OTAN, obtained from a real flying drone [23]. The data have been originally provided in the context of a challenge, where the participants' task was to track, classify, and identify Class I UAs as they fly 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



(a) Performance of the PPID protocol.



(b) Performance of the e-PPID protocol.

Figure 4.4: Execution time of the runtime phase of the protocol, with various ECC curves.

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 an invasion, if any.

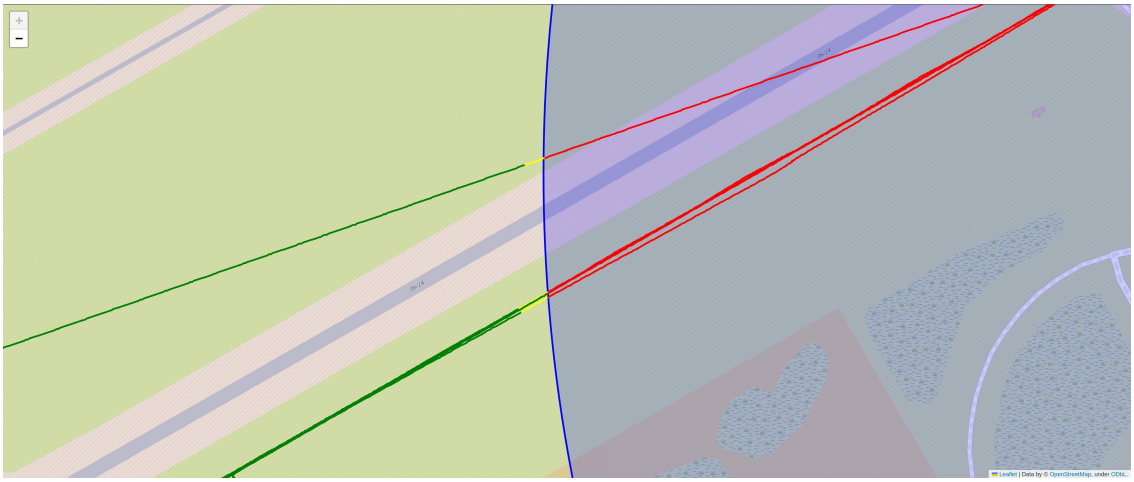


Figure 4.5: Invasion detection of a drone during its flight. The path is green (no invasions) when outside of the circle, yellow when close to the circle (future invasion), and red (invasion detected) when inside the circle.

Figure 4.5, shows the safe (i.e., non-invading) flight path of the drone in green, with the protocol detecting the future and current invasion, in yellow and red, respectively. This simulation allowed us to understand the behavior of the protocol and its capabilities of correctly predicting future invasions. It is worth noting that an invasion of the current location has higher precedence than a future invasion, meaning that when a drone is invading, we do not care about the future invasion. 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.

- **Simulation on Android.** In order to bring our protocol to life in a real-world context, we've developed an Android application. Given that our protocol is written in C, we leveraged Android's native C compatibility to our advantage. The choice of using the Android for simulation comes from the fact that a typical android smartphone has all the necessary features and hardware which are required by the drone, and it is easily available and programmable. Note that, this simulation on android is not to measure the real-time performance of the drone but to measure the correctness of the protocol.

Communication Cost

The communication cost plays an important role in the performance because the drone might have to respond to several CIs at the same time. Thus, higher communication costs would

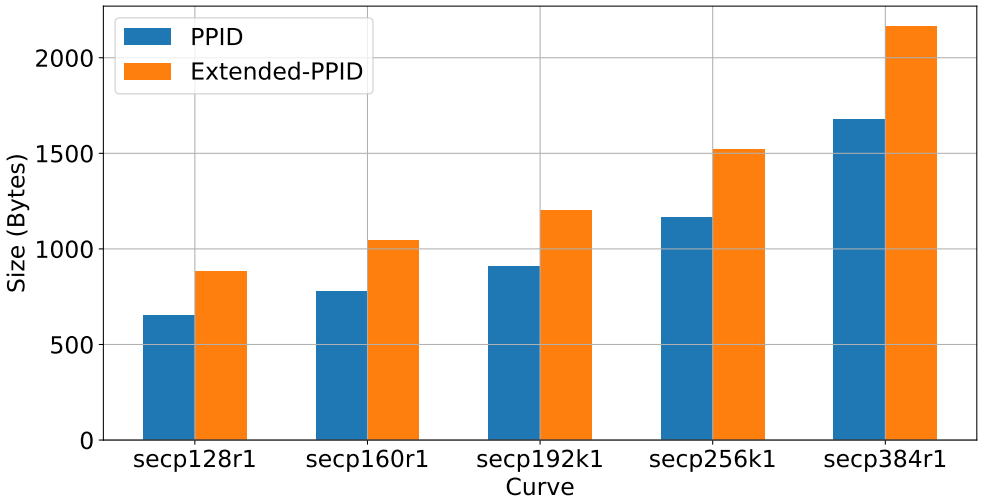
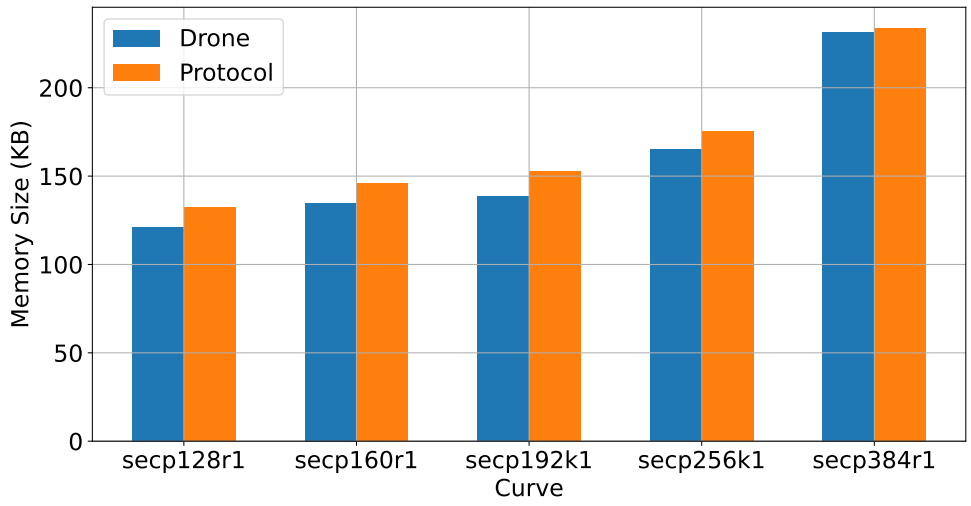


Figure 4.6: Total bandwidth (RX + TX) used by the drone for both protocols.

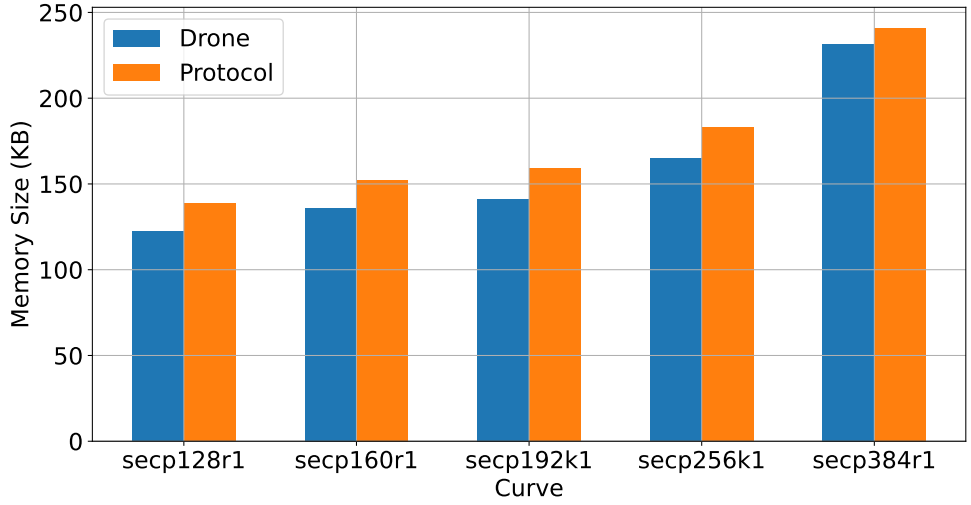
reduce the available bandwidth and hence cause delays that may impact drones operations negatively. Figure 4.6 shows the communication cost of our protocols for different key sizes. We see that the e-PPID protocol requires higher bandwidth compared to PPID, as it requires additional messages to be delivered and a slightly bigger message size, due to AES encryption of the future location. Communication costs do not include the size of the origin and radius, which is variable and depends on the protocol setup and the precision of the coordinates.

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 4.7 reports a comparison of 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. It is visible that the difference between the drone and protocol memory 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.



(a) PPID Protocol.



(b) e-PPID Protocol.

Figure 4.7: Memory overhead required by the protocols, with various ECC curve sizes.

5
Related Work

In this section, we revise the literature related to our work on drone location privacy and privacy-preserving collision avoidance and testing.

Drone Location Privacy. Svaigen et al. considered the location privacy issue in the internet of drones [24]. However, their solution mostly focuses on the anonymity of drones rather than location privacy. In [25], 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 [26], where the authors consider the problem of location privacy where the drone collaborates with a ground vehicle to save energy while traveling. 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 [10]. The authors proposed the use of 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 [27, 28] and co-presence verification [29]. 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 [14] 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 UAVs 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 [14] to the problem tackled in this manuscript would not allow any entity to disclose the location of the invading drone, being useless for the application of the following countermeasures.

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 [18]. On the one hand, we notice that the straightforward application of such a primitive in this context would not allow to address our problem, as the cited solution is 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., [30] and [31], to name a few. However, PPID builds on top of the solution in [18] 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

In this paper, we proposed PPID, a protocol allowing Critical Infrastructure (CI) operators to detect unmanned aircraft (drones) inadvertently invading no-fly areas, while preserving the location privacy of the vehicles. We also proposed e-PPID, a variation of PPID allowing to discover future invasions, while still preserving drones' location privacy. We verified the logic security of our protocols through the automated tool *ProVerif*, and we also implemented them in an actual proof-of-concept using a constrained device. Our extensive performance assessment shows that PPID achieves private invasion detection by requiring only 52.31 msec when configured to operate on the curve *secp*256*k*1, being definitely supportable by commercial drones. Future work will consider further optimizations of the protocols, as well as the detection of potentially malicious drones, using forged locations.

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] BBC, "Gatwick Airport: Drone attack grounds flights," http://www.bbc.co.uk/news/uk-england-sussex-4662375, 2019, accessed: 2022-03-21.

[7] 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: 2022-03-21.

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

[9] FAA, "Remote identification of unmanned aircraft," https://www.faa.gov/news/media/attachments/RemoteID_Final_Rule.pdf, 2021, accessed: 2022-03-21.

[10] A. Brighente, M. Conti, and S. Sciancalepore, "Hide and seek: Privacy-preserving and faa-compliant drones location tracing," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–11.

[11] 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: 2022-03-21.

[12] X. Chang, C. Yang, J. Wu, X. Shi, and Z. Shi, "A surveillance system for drone localization and tracking using acoustic arrays," in *2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2018, pp. 573–577.

[13] 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.

[14] 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.

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

[16] V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila, "A survey of the elliptic curve integrated encryption scheme," 2010.

[17] 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.

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

[19] 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.

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

[21] S. Sciancalepore, A. Brighente, M. Conti, and H. Vaishnav, "Source code of PPID into ProVerif," https://github.com/harshul-vaishnav/ppid, 2023.

[22] D. R. George, S. Sciancalepore, and N. Zannone, "Privacy-Preserving Multi-Party Access Control for Third-Party UAV Services," in *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 19–30.

[23] NATO, "Drone identification and tracking," https://www.kaggle.com/c/icmcis-drone-tracking/overview, 2021, accessed: 2022-03-21).

[24] 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.

[25] ——, "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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

Acknowledgments

So, we've made it to the finish line. I couldn't have done it without some great people by my side.

First up, a big thank you to my supervisor, Prof. Alessandro Brighente. You've been my guide through all this, always ready with advice, support, and a bit of wisdom when I needed it. I wouldn't have gotten far without you.

My co-supervisor, Prof. Savio Sciancalepore, thank you for broadening my perspectives and making my work stronger. You always challenged me to think deeper and your input was absolutely invaluable.

A special shout out to my thesis mentor, Prof. Mauro Conti. You've been there for me beyond just the academic stuff, cheering me on, and helping me keep going when things got tough. I can't express enough how grateful I am for your support.

I'd also like to express my gratitude to University of Padova. The environment, resources, and the opportunity to learn from some of the best minds have been invaluable to my academic journey. To my coursework professors, your teaching has not only enriched my knowledge base but also helped me develop a critical thinking approach that was crucial while working on this thesis.

Can't forget my family and friends, who have put up with me during this time. Thanks for always being there, for understanding when I was stressed, and for cheering me on.

So, to all of you, thank you. This thesis may bear my name, but it's a testament to the collective effort and support I've received from each one of you. I am immensely grateful.