Privadome: Delivery Drones and Citizen Privacy

Gokulnath M. Pillai*, Ajith Suresh†, Eikansh Gupta‡, Vinod Ganapathy, Arpita Patra
Indian Institute of Science, Bangalore *Cerebras, Bangalore
†Technology Innovation Institute, Abu Dhabi ‡Qualcomm, Hyderabad

ABSTRACT

E-commerce companies are actively considering the use of delivery drones for customer fulfillment, leading to growing concerns around citizen privacy. Drones are equipped with cameras, and the video feed from these cameras is often required as part of routine navigation, be it for semi-autonomous or fully-autonomous drones. Footage of ground-based citizens captured in these videos may lead to privacy concerns. This paper presents Privadome, a system that implements the vision of a virtual privacy dome centered around the citizen. Privadome is designed to be integrated with city-scale regulatory authorities that oversee delivery drone operations and realizes this vision through two components, Pd-Mpc and Pd-Ros. Pd-Mpc allows citizens equipped with a mobile device to identify drones that have captured their footage. It uses secure two-party computation to achieve this goal without compromising the privacy of the citizen's location. Pd-Ros allows the citizen to communicate with such drones and obtain an audit trail showing how the drone uses their footage and determine if privacy-preserving steps are taken to sanitize the footage.

1 INTRODUCTION

This paper concerns the problem of citizen privacy in the era of delivery drones. Prior studies [24, 41, 69, 96, 107] have shown that citizens perceive drones as a threat to their privacy, and rightly so. Drones are equipped with cameras for navigation, and the video feeds captured by these cameras may record footage of ground-based citizens and their private spaces [104]. A recent survey in the US found that 88% of the participants were concerned about delivery drones recording their footage and using it for marketing and advertising [31]. Admittedly, the delivery drone sector is regulated by government oversight (*e.g.,* the identities of drones are generally known), and the drones belong to large e-commerce companies with reputations to protect. Nevertheless, privacy remains a problem because citizens have no way to reason about how the footage is stored or used. Recent drone-based food and coffee delivery trials by Alphabet Wing in various Australian cities led to citizen concerns and a parliamentary report calling for oversight on privacy [25, 73]. European drone vendors recently formed Drones4Sec [42], a body to define protection of personal data in the era of drones (among other objectives). Laws proposed by various countries have suggested that data gatherers (*e.g.,* drone operators) must incorporate suitable accountability mechanisms to protect citizen privacy [77]. Large e-commerce companies are often asked by governments to explain how they use the collected data [37].

In this paper, we propose a framework called **Privadome** to protect citizen privacy in the presence of delivery drones. Privadome aims to implement the vision of a *virtual privacy dome*, centred around the citizen, that protects their privacy from the prying cameras of delivery drones. Privadome detects a delivery drone (or multiple drones) that may be in the vicinity of a citizen and determines whether that citizen is captured in the field of view of the drone's camera. Privadome also provides mechanisms for the citizen to determine whether the pictures/video captured by the drone's camera are suitably sanitized to protect privacy, *e.g.,* that all faces captured in the video feed are blurred (as is done in Google Street View). From the perspective of a ground-based citizen, Privadome simply requires the citizen to install an application on their mobile phone. This mobile application helps determine the citizen's location, which is then used to identify drones in the citizen's vicinity and start the workflow in Privadome.

Privadome is designed to be integrated with a city-scale regulatory authority that oversees delivery drone operations. These regulatory authorities are region-specific, *e.g.,* the Federal Aviation Authority in the US [13], the Civil Aviation Authority in the UK [12], or the Directorate General of Civil Aviation in France and in India [53, 54]. The regulatory authority must be aware of the identity and current location of each delivery drone operating in the city. This assumption is not just realistic for the delivery drone sector, but in fact a requirement, to ensure collision-free delivery routes. Such requirements have been proposed in the drone laws of various countries (*e.g.,* USA [13], France [101], EU [47], Switzerland [34], and India [54]), and drone security vendors are offering deployable tracking solutions [86, 93].

Privadome has two components: Pd-Mpc, which we describe next, and Pd-Ros, an auditing framework for the citizen to determine if privacy is maintained in the recorded footage. **Pd-Mpc** allows a citizen to identify delivery drones whose cameras have the citizen in their field of view (§3). Pd-Mpc uses secure multiparty computation (MPC) between the regulatory authority's server and the citizen's mobile phone to accomplish this goal *without revealing the citizen's location.* Two-party MPC, which we use, enables a pair of mutually-distrusting parties to collaboratively compute a function without revealing anything besides the function output. Pd-Mpc encodes a geometric computation that incorporates each drone's location, its direction of motion, the specifics of the drone's camera hardware and the citizen's location. This geometric computation determines if the citizen appears anywhere in the camera's field of view. Pd-Mpc's use of MPC ensures that it is able to accomplish this goal without requiring the regulatory authority to reveal the locations of all the drones in the city to the citizen.

However, naïve use of MPC in this setting presents scalability problems. Regulatory authorities operate at a city-scale, possibly

Proceedings on Privacy Enhancing Technologies 2024(2), 29–48
© 2024 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2024-0039

tracking hundreds of delivery drones at any given time. Several thousand citizens may also simultaneously query the regulatory authority to identify drones in their vicinity. Off-the-shelf MPC protocols are computationally expensive and will have difficulty operating with the number of drones that one may expect at a city-scale. Moreover, each citizen communicates with the regulatory authority with their mobile phone, and traditional MPC protocols consume significant network bandwidth. Pd-Mpc is carefully engineered to scale to hundreds of drones, making it suitable for near-term city-scale deployments. Our evaluation of Pd-Mpc shows that each query from the citizen (to check their privacy at city-scale) only consumes up to 6.59MB of mobile data for city-scale deployments of up to a thousand delivery drones. Note that the regulatory authority can execute queries from different citizens in parallel, and therefore automatically scales to an arbitrary number of citizen queries using replication.

Once a citizen identifies a drone(s) that has captured their footage, they may wish to ensure that their privacy is maintained in the recorded footage. This requires communicating with the drone, either directly or via the regulatory authority. A well-intentioned delivery drone must then convince the citizen that the video has been sanitized appropriately. However, the citizen must have mechanisms to trust the drone's assertions that the data is sanitized.

Pd-Ros is an exemplar framework aimed to provide such assurances on ROS2-based drones (§4). ROS2—the Robot Operating System, version 2 [83]—is a popular middleware used in the software stacks of drones by numerous vendors. Pd-Ros enhances ROS2 to audit data flows between applications within the drone. Pd-Ros relies on trusted hardware on the delivery drone to provide the citizen with an audit trail of how their data is used within the drone. Many regulatory bodies do require delivery drones to be equipped with such trusted hardware (see §2 for references).

Prior related methods in this area have been tailored toward ground-controlled drones [17, 18, 67, 68]. They aim to detect if the drone has captured a citizen's footage in the first-person view exported to a ground-based human operator. Privadome's methods are agnostic to whether the drone is ground-controlled or fully-autonomous. Companies such as Amazon are considering using fully-autonomous drones for their delivery fleets [5], and prior methods will not work with such drones. Privadome is the first system with mechanisms for citizens to obtain an audit trail from a drone that has captured their footage to maintain their privacy.

At this point, it is natural for the reader to question our choice of focusing exclusively on the delivery drone sector. After all, citizen privacy can just as well be compromised by end-user drones that are not delivery drones, so what purpose does a delivery drone domain-specific system like Privadome serve? We answer these concerns with three propositions.

• First, *the delivery drone sector is commercially important.* Many e-commerce giants have set up drone delivery units (*e.g.,* Google Wing, Amazon Prime Air), and the economic and logistical advantages of drone-based delivery are beyond question. The technology has also sufficiently advanced, as evidenced by successful recent trials. Privacy concerns, however, remain a significant hurdle to the wide-spread adoption of delivery drones [24, 30, 41, 69, 73, 96, 107]. Privadome, tailored specifically to this sector, is therefore likely to be of significant interest to the community.

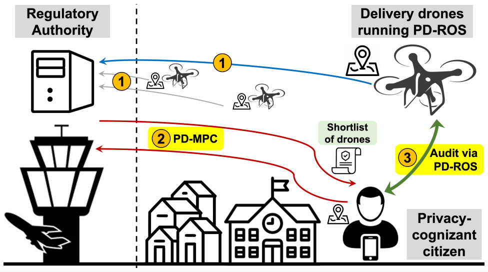
• Second, *a system like Privadome can in fact encourage adoption of delivery drones.* Trials of delivery drones in the recent past have raised concerns from privacy-conscious citizens [25, 73]. Studies [66, 100] indicate that privacy concerns by citizens can in fact impact technology uptake and sales, with a study by Cisco [27] showing 90% of customers would not buy from companies that do not adequately protect their data privacy. Indeed, privacy disclosures by delivery drone services of e-commerce giants do not inspire much confidence. For example, Google Wing's privacy policy states that "the cameras on Wing's drone may incidentally capture low-resolution black-and-white overhead images . . . which may include images of individuals" [103], while Amazon's drone delivery FAQ mentioned that ". . . the cameras may record overhead videos of people and things near your delivery location when completing the delivery process.[1]" We therefore believe that the adoption of a Privadome-like solution, which offers audit trails attested by trusted hardware, can build citizen confidence and encourage the uptake of drone-based delivery.

• And third, *the presence of regulations in this sector makes a solution like Privadome technologically feasible.* Delivery drones are operated on behalf of e-commerce companies that have reputations to protect, and their operations are overseen by a regulatory authority. Delivery drones are required to declare their location and are often equipped with trusted hardware. There is an incentive to comply with local laws and regulations, and a system like Privadome can enable enforcement of citizen privacy on these drones.

In fact, regulatory frameworks such as the European Union's General Data Protection Regulation (GDPR), California's Consumer Protection Act (CCPA), and India's recent Digital Personal Data Protection Bill (2023) explicitly call for data gatherers to provide accountability mechanisms that allow end-users to determine what data related to them has been gathered and request deletion if required. Privadome's mechanisms allow end-users to rigorously determine that any delivery drone image/video feeds that have captured their footage have been suitably sanitized. We therefore believe that Privadome-like mechanisms will assist delivery drone operators to adhere to privacy regulations in GDPR, CCPA and other regulatory frameworks. Some privacy issues posed by delivery drones may exceed the scope of these regulatory frameworks. The image/video feed may gather the footage of a wide variety of people, many of whom may simply be bystanders who are not even be registered with the e-commerce operator that owns the delivery drone. Regulatory frameworks like GDPR only obligate data gatherers to protect private data of registered end-users. However, Privadome's mechanisms can protect the privacy of *all* users captured in the drone's feed.

Admittedly, it is challenging to protect citizen privacy in the presence of rogue end-user drones (*i.e.,* in the unregulated sector). Solutions to detect citizen privacy violations by unregulated drones have only been developed for certain kinds of drones [18, 67]. These solutions also involve active participation from the citizen, *e.g.,* to

[1] We obtained this statement verbatim from Amazon's drone delivery FAQ circa May 2023 [8], but as of September 2023, the same FAQ page [7] no longer has any mention of privacy for drone-based deliveries. The removal of this rather important privacy-related statement from the FAQ further justifies our belief that privacy policies in the delivery drone sector are evolving and very much a work-in-progress. This does little to inspire confidence in privacy-conscious citizens, further justifying the need for Privadome-like approaches.



① Delivery drones register with a city-scale regulatory authority that oversees their operation. Each drone has a unique identifier and keeps the regulatory authority updated with its location. ② Privacy-conscious citizen queries the regulatory authority with their current location (using PD-MPC), and obtains a shortlist of drones whose cameras may have the citizen in their field of view. ③ Citizen communicates with the drone, either directly or via the regulatory authority (without revealing their identity). Well-intentioned drones running PD-ROS can provide an audit trail to citizens showing that their privacy is maintained in the footage.

Figure 1: Deployment vision for Privadome.

perturb an object/person to see if it is in the field of view of a drone [67] or to install WiFi detectors on home windows to detect drones hovering outside [18]. A general solution to address citizen privacy in the presence of unregulated drones will require a legal framework and law-enforcement methods, both of which are actively being considered [82], but outside this paper's scope.

Unregulated drones will continue to compromise citizen privacy for the foreseeable future, until further regulations are in place. However, we also believe that this fact must not preclude the development of a privacy solution in the commercially-important delivery drone sector. This belief led us to build Privadome.

To summarize, Privadome's key contributions are:

- PD-MPC, an MPC-based system for citizens to identify drones that have captured their footage while preserving location privacy of citizens. PD-MPC is carefully engineered to minimize network communication on citizens' mobile devices and work with city-scale drone deployments. Our evaluation simulating various city-scale deployments shows that for near-term deployments (∼1000 drones), PD-MPC consumes lesser network bandwidth on the citizen's mobile device than streaming low-resolution YouTube videos.

- PD-ROS, an auditing framework on ROS2-based drones, for citizens to determine if their captured footage is suitably sanitized. PD-ROS' workflow consumes a modest amount of additional CPU resources and power when implemented on a NVidia Jetson Xavier NX development board, with hardware similar to those on drones.

2 OVERVIEW AND THREAT MODEL

Figure 1 presents the envisioned deployment scenario and overall workflow of Privadome. Privadome is designed to be integrated with a wide-area (*e.g.,* city-scale) regulatory authority with oversight on delivery drone operations. We assume that the identity of each drone is known to the regulatory authority and that the drone provides real-time location updates to the regulatory authority (**Step ①** in Figure 1). Indeed, such requirements have been stated in the drone laws of various countries. For instance, the US Federal Aviation Authority (FAA) has proposed that all delivery drone operations from 2023 must comply with remote identification rules, which require each drone to update the FAA with its identity, current location, altitude and velocity, in addition to other required data [13]. France [101], the European Union [47], Switzerland [34], Australia [72, 73] and India [54] have also proposed similar regulations for drone operations. Some countries (*e.g.,* India [54]) even require drones to declare their flight path and seek permission from the aviation authority prior to take-off. Security vendors offer commercial trackers that drones can use for regulatory compliance [93].

Privadome additionally requires that the particulars of the drone's camera system (its focal length and sensor size) be known to the regulatory authority. This information is used in the geometric computation in PD-MPC. Although current regulations do not explicitly state that drones must make these details known to the regulatory authority, we feel that this is a reasonable assumption. These details are generally publicly available and associated with the drone's make and model.

When a citizen wishes to determine if a drone is recording their footage, they query the regulatory authority using an application on their mobile phone (**Step ②** in Figure 1). The application can be configured to send such queries periodically. This application allows the user to specify a *vicinity radius* centred around them. The regulatory authority will detect drones within the vicinity radius of the citizen that have captured the citizen's footage. Our focus in this paper is on the privacy of the citizen's footage itself. However, this application can easily be extended so that the citizen can also specify other locations (*e.g.,* their home or yard) whose privacy they would like to protect in the footage recorded by delivery drones.

In Privadome, citizens interact with the regulatory authority using PD-MPC, which uses a two-party MPC protocol. Its inputs are the details of all the drones in the city (the location, direction of motion, and camera particulars of each drone), as provided by the regulatory authority, and the citizen's location. PD-MPC uses this information to output to the citizen the identities of drones that may have captured the citizen's footage. PD-MPC goes beyond the traditional step of simply identifying drones in the citizen's vicinity (in fact, some drone-installed trackers already wirelessly broadcast their presence to nearby devices [86]). PD-MPC uses the direction of motion of each drone and the details of the drone's camera in a geometric computation to determine whether the citizen's location is within the field of view of the drone's camera. Only such drones are shortlisted in the output of PD-MPC. As delivery drone operations begin to increase in popularity, we can expect several tens of drones to be present within typical vicinity radii that citizens may specify (*e.g.,* 500 metres). By shortlisting drones that have the citizen in their field of view, PD-MPC reduces the number of drones from which the citizen must request an audit trail in Step ③.

PD-MPC uses MPC to accomplish this goal without revealing the citizen's location to the regulatory authority. This is important because even though the regulatory authority is likely to be a trustworthy body, citizens will find it unpalatable to reveal their locations. Even if the citizen's queries anonymously reveal location, *i.e.,* without disclosing the citizen's identity, prior work shows that citizens can be de-identified using historical query data (*e.g.,* [76]).

PD-MPC encodes the entire aforementioned geometric computation using the machinery of MPC, thereby ensuring the citizen's location privacy. In that sense, Privadome views the regulatory authority as an honest-but-curious participant. Delivery drone operators may also not wish to make the location of all their delivery drones publicly available to anybody that poses a query to the regulatory authority. PD-MPC has the pleasant side-effect of also ensuring this goal (to an extent), but this is a not our primary goal, whereas protecting citizen privacy is. The identities of drones that are in the citizen's vicinity are revealed to the citizen.

With the shortlist of drones in hand, the citizen can then request each delivery drone to show that it is taking measures to ensure privacy in the recorded footage (**Step ③**). This communication between the citizen and the drone can either happen directly, assuming supporting infrastructure for such direct communication exists, or mediated by the regulatory authority. This communication happens anonymously, without revealing the citizen's identity either to the drone or the regulatory authority. Each drone can then provide suitable proof to show that it is complying with region-specific privacy laws. For example, it could present an audit trail to the citizen showing that the footage is being sanitized on-board the drone or that the footage is being recorded at low resolution.

Modern drones routinely capture video footage that is used to detect and avoid obstacles during navigation. Thus, it is impossible to offer a solution in which sensitive objects or people do not get captured in the video footage. Prior solutions [67] that simply detect whether a citizen/object appears in the video footage will therefore flag (as suspicious) drones that may otherwise be well-intentioned, and happened to capture the footage of the citizen in their field of view during routine navigation. Privadome's approach is to provide a framework (namely, PD-ROS) that allows well-intentioned drones to provide an audit trail to citizens that their privacy is being preserved in the video footage. Drones that cannot provide a satisfactory audit trail or fail to communicate with the citizen can be reported (identities of shortlisted delivery drones are known to the citizen and the regulatory authority). Privadome can thus abate citizen privacy concerns, such as those raised after a recent Alphabet Wing drone delivery trial [25], which in turn incentivizes drone delivery companies to adopt a Privadome-like solution.

PD-ROS is an auditing framework for ROS2-based drones. We chose to demonstrate our approach on ROS2 because of its popularity among drone vendors, *e.g.,* various models sold by DJI, 3DR, Parrot, Gaitech, Erle, BitCraze, and Skybotix use ROS2. It should be possible to develop PD-ROS-like solutions for other drone software stacks as well. In our PD-ROS prototype, we ensure privacy by checking that applications on the drone only consume video footage from the camera after it has been sanitized, *e.g.,* to blur faces that appear in each frame. ROS2 is a publish/subscribe system, in which applications publish and subscribe to *topics*, *e.g.,* the camera may publish to a topic called *VideoFeed*, to which the navigation application may subscribe. ROS2 sets up communication between applications by matching topics, and applications declare the topics to which they publish or subscribe in a manifest. Audit trails record the manifest of the application when it is launched. Manifests show how applications are permitted to communicate. Citizens can use them to verify that raw video footage is sanitized for privacy before being consumed by downstream applications.

The key challenge, however, is to provide a basis for the citizen to establish trust in the integrity of the audit trail presented by a delivery drone. In Privadome, we address this problem by requiring delivery drones to be equipped with trusted hardware. We use the ARM TrustZone [10] in our experimental prototype, but any similar attestation hardware should suffice. PD-ROS uses the ARM TrustZone to: ① run the trusted software that sanitizes image feeds; ② procure and securely store the data producer/consumer information (the audit trail); and ③ digitally signs and sends the audit trail to a citizen that requests a proof of footage sanitization.

It is natural to ask whether requiring delivery drones to be equipped with trusted hardware overly restricts the scope of Privadome. In response, we note that drone laws are beginning to recognize and provide special certifications to drones that are equipped with trusted hardware. For example, India offers the higher-grade "Level 1" certification only to drones that have a hardware-backed trusted execution environment [54]. Drone vendors are also beginning to incorporate trusted hardware capable of securely storing information and performing cryptographic operations, *e.g.,* the Wisekey Secure Element on the Parrot ANAFI Ai drone [74].

That said, laws governing delivery drone operations are still evolving in various countries, and it is unclear if all countries will require delivery drones to be equipped with trusted hardware. For example, they may simply require that the drone be equipped with a certified software stack, but not require any hardware root of trust on the drone (*e.g.,* the lower-grade "Level 0" certification given to drones in India [54]). Because PD-ROS is a set of tools atop the drone software stack, we expect that these tools can also be used on (ROS2-based) drones that lack trusted hardware. However, in the absence of trusted hardware the guarantees provided will be correspondingly weaker. For example, the audit trail cannot be digitally signed by the hardware root-of-trust, and the citizen will have to trust that the software stack on the drone is untampered to establish the integrity of the audit trail.

Threat Model. Because Privadome's main goal is to protect citizen privacy, its threat model is citizen-centric. The regulatory authority is assumed to be honest-but-curious. It truthfully engages with citizens to identify drones that have their footage, but citizens do not have to reveal their location to the regulatory authority. The regulatory authority must be able to track the locations of all delivery drones registered with it. We expect regulatory authorities to be operated by a trusted entity (*e.g.,* the government) and therefore rule out the possibility of collusion between the regulatory authority and delivery drone operators.

We assume that delivery drones are owned by large e-commerce companies with reputations to protect. However, Privadome only trusts these e-commerce companies to the extent that it expects them to operate drones equipped with a trusted execution environment, such as the ARM TrustZone. It expects the drones' identities and details of their camera hardware to be registered with the regulatory authority. The public key associated with the trusted hardware can itself serve as the drone's identifier. We assume the existence of supporting public-key infrastructure, *i.e.,* a certifying authority that issues digital certificates for the public keys of registered drones; the regulatory authority can serve this role. For Privadome, the trusted-computing base on the drone is just the trusted hardware, capable of securely storing the audit trail and attesting the software stack on the drone. Note that e-commerce companies have historically relied on decentralized delivery fleet management. They often engage the services of *third-party delivery service fleet operators (DSPs)* (*e.g.,* see [28, 43, 46, 62]), who procure and operate the delivery vehicles. While we can reasonably assume that e-commerce giants have no overt intention to break local privacy laws, the same cannot be assumed of DSPs. Indeed, Privadome's threat model *does not trust DSPs*, but does assume that the e-commerce company requires DSPs to operate drones equipped with trusted hardware, which in turn can attest that the DSP has not tampered with the drone's software stack.

One could ask why Privadome-like mechanism is needed at all, if the e-commerce companies are assumed to be benign in intent. The answer is that it is still important to have accountability mechanisms in place to ensure that the e-commerce companies are abiding with local privacy laws [77]. Having accountability mechanisms such as Privadome both acts (1) as a deterrent to the drone operators from violating local laws; and (2) as a confidence building means for citizens, who in turn will be more open to adopting delivery drone-based services. Indeed, history has shown that e-commerce giants are sometimes caught violating privacy of their clients [37, 44, 90, 102]. This underscores the need for a deterrent and an accountability mechanism available to citizens.

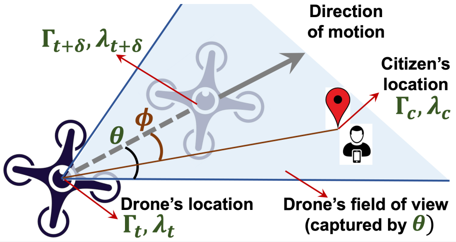
Privadome's threat model excludes drones that are not registered with the regulatory authority and other rogue drones that deliberately try to hide their current location. We also exclude from our threat model drones that use attached cameras that are outside of the purview of the drone's attested software stack. For example, a rogue drone operator can attempt to bypass Privadome by physically attaching a Go-Pro camera to the drone. Neither the trusted hardware on the drone nor the ROS2 software stack will have any control over such an attached camera. Other methods and regulations are required to deter such attacks. We expect that e-commerce giants will not engage in deliberate attempts to violate local privacy laws because they have reputations to protect. However, individual DSPs may engage in such attacks. E-commerce giants can deter DSPs from engaging in such attacks by requiring the DSP to provide a live photograph of the drone prior to take-off from the warehouse, which can subsequently be examined for the presence of unauthorized attachments. Finally, several models of drones can either themselves record audio or carry peripherals to do so. End-user privacy can also be violated via recorded audios, but the methods in this paper are tailored for video alone.

3 SHORTLISTING DRONES WITH Pd-Mpc

Pd-Mpc is a system based on two-party MPC, involving a citizen and the regulatory authority. At the end of Pd-Mpc, the citizen has a short list of drones in whose field of view he/she appears. We first present the geometric computation used to detect if the citizen is in the drone camera's field of view (§3.1), describe how to encode this computation in an MPC framework (§3.2), present implementation details (§3.3) and an evaluation of our Pd-Mpc prototype (§3.4).

3.1 Drone Shortlisting Algorithm

Figure 2 depicts the basic setup of the geometric computation. For simplicity, this figure considers a bird's eye view in two dimensions, with both the drone and the citizen on the same plane. Suppose we assume that a drone is within the citizen's vicinity radius, and that its location at a given instant in time is given by the pair of GPS coordinates (Γ_t, λ_t). To determine whether the citizen at (Γ_c, λ_c) is within its field of view, we need to know both the cone of vision of the drone's camera and the direction in which the drone is moving (assume for now a fixed, forward-facing camera, which we subsequently relax). The regulatory authority can compute the angle of the cone of vision (θ in Figure 2) using the details of the drone's hardware with a well-known formula [64]: $\theta = \texttt{arctan}(d/2f)$, where f is the focal length of the camera's lens and d is the dimension of the sensor (*e.g.,* the CCD sensor) used to digitally record the image. These parameters are typically associated with the make and model of the drone, which we assume is known to the regulatory authority. Algorithm 1 therefore simply uses θ as an input.



We use the drone's current GPS location (Γ_t, λ_t) and location after a short time interval ($\Gamma_{t+\delta}$, $\lambda_{t+\delta}$) to determine its direction of motion. The drone camera's field of view is abstracted by the parameter θ. Pd-Mpc's geometric computation determines whether the citizen's location (Γ_c, λ_c) is within the drone's field of view, *i.e.,* whether $\phi < \theta$. For the field of view computation, Pd-Mpc converts location information captured as GPS coordinates into an equirectangular projection with (Γ_t, λ_t) as origin.

Figure 2: Setup for Pd-Mpc's geometric computation.

The regulatory authority captures the drone's direction of motion using its GPS coordinates after a short interval of time, shown as ($\Gamma_{t+\delta}$, $\lambda_{t+\delta}$) in Figure 2. The citizen lies within the camera's field of view if the angle shown as ϕ in Figure 2 is less than θ. On a two-dimensional plane with (Γ_t, λ_t) as the origin, ϕ is the angle between the two vectors denoting the drone's direction of motion ($\overrightarrow{\mathbf{D}}$) and the citizen's location with respect to the origin ($\overrightarrow{\mathbf{C}}$). The value of ϕ can be determined as shown in line 5 of Algorithm 1. Recall that locations are reported as GPS coordinates, which are in latitudes and longitudes. To obtain the vectors $\overrightarrow{\mathbf{D}}$ and $\overrightarrow{\mathbf{C}}$, we need to obtain equirectangular projections of the GPS coordinates [88] with (Γ_t, λ_t) as the origin. Vectorize, called on lines 3 and 4 of Algorithm 1 accomplishes this task. A few observations about Algorithm 1:

- *Vertical Field of View.* The algorithm only considers the horizontal field of view, and ignores the vertical field of view. A citizen lies within the vertical field of view of the drone if $\phi_v < \theta$, where $\phi_v = \texttt{arctan}(h/\text{Dist})$, where h is the altitude of the drone, and Dist is the distance between the drone and the citizen, as calculated on line 1. Although this calculation is simple enough to include in the algorithm, we chose not to do so because of two reasons. First, it is reasonable assume that a citizen appears within the vertical field of view of a drone within the vicinity radius. The citizen will likely be out of view only if the drone is too close to the citizen, *e.g.,* hovering right overhead, in which case $\phi_v > \theta$). For drones that are in such proximity, we assume that the citizen's footage would have been captured on the drone's approach path, and that the citizen would be interested in obtaining an audit trail from that drone anyway. If the drone is equipped with an optical flow camera (usually downward facing) it will capture the citizen's footage even if it is hovering right overhead. Second, the vertical field of view computation involves a division (h/Dist) and a trigonometric ($\texttt{arctan}$) operation, both of which are expensive in MPC. Even precisely computing Dist is computationally expensive in MPC (§3.2).

Algorithm 1: DetectFieldOfView.

```
Input: From citizen: $\Gamma_c$, $\lambda_c$, VicinityRadius.
Input: From regulatory authority: $\Gamma_t$, $\lambda_t$, $\Gamma_{t+\delta}$, $\lambda_{t+\delta}$, $\theta$.
Output: True if citizen in field of view, else False.
// Each ($\Gamma$, $\lambda$) is a GPS latitude/longitude pair.
1  Dist = Distance($\Gamma_c$, $\lambda_c$, $\Gamma_t$, $\lambda_t$)
2  if (Dist > VicinityRadius) then return False
3  $\overrightarrow{\mathrm{D}}$ = Vectorize($\Gamma_t$, $\lambda_t$, $\Gamma_{t+\delta}$, $\lambda_{t+\delta}$)
4  $\overrightarrow{\mathrm{C}}$ = Vectorize($\Gamma_t$, $\lambda_t$, $\Gamma_c$, $\lambda_c$)
5  $\phi$ = arccos(($\overrightarrow{\mathrm{D}} \cdot \overrightarrow{\mathrm{C}}$) / ($|\overrightarrow{\mathrm{D}}| \times |\overrightarrow{\mathrm{C}}|$))
6  if ($\phi \leq \theta$) then return True else return False
7  Procedure Vectorize (Snyder [88])
       Input: GPS coords of Origin & Target: ($\Gamma_o$, $\lambda_o$, $\Gamma_t$, $\lambda_t$).
       Output: Vector denoting Target in an equirectangular projection with
               Origin as origin.
8      X = $\mathbf{R} \times (\lambda_t - \lambda_o) \times \cos(\Gamma_o \times \frac{\pi}{180})$
9      Y = $\mathbf{R} \times (\Gamma_t - \Gamma_o)$                  // R = Earth's radius
10     return the vector [X, Y]
```

- *Gimbal-mounted Cameras/Multiple Cameras.* The algorithm assumed a fixed camera, facing forward in the direction of motion of the drone. On drones with a gimbal-mounted camera, the camera may not necessarily point forwards in the direction of the drone's motion. Algorithm 1 can easily incorporate drones with such cameras. The drone will have to additionally communicate the pitch and yaw of the gimbal to the regulatory authority. Using this information, the regulatory authority can calculate the angle (α) between the drone's direction of motion and the camera's orientation. The angle α can be applied as a corrective factor to ϕ on line 5. The same idea can also be applied to account for the yaw and pitch of the drone itself, and suitable corrective factors can be applied to ϕ. The algorithm can also accommodate drones that are equipped with multiple cameras by considering a consolidated cone of vision that encompasses the cones of vision of each individual camera.
- *Camera Resolution.* The algorithm requires the citizen to provide a vicinity radius as input (typically a few 100 meters); only drones within this vicinity radius are considered. However, cameras differ in their resolution, with higher-resolution cameras capable of capturing sharp images from afar. Using a single vicinity radius ignores information about camera resolution. However, Algorithm 1 can incorporate this information if the regulatory authority itself suggests suitable vicinity radii, based on the resolution of the drone's camera (which it can determine from the drone's model). In this case, citizens specify the maximum acceptable resolution at which they are comfortable being captured in the footage, and the regulatory authority suggests a suitable vicinity radius for each drone.

3.2 Encoding as an MPC Computation

Pd-Mpc encodes DetectFieldOfView as a secure two-party computation between the citizen and the regulatory authority. However, Algorithm 1 requires a few modifications to make it secure and efficient for use in an MPC framework. We detail those modifications here, but first provide some background on MPC frameworks.

Broadly, there are two popular regimes of MPC protocols. One uses Yao garbled circuits (GC) [106]. In GC, the circuit garbler generates the garbled circuit and gives it to the other party who evaluates it. The circuit evaluator obtains the other party's inputs using oblivious transfer (OT) before evaluating the circuit, and the rest of the evaluation is non-interactive. The circuit evaluator then reveals the values at the output gate to the other party. In GC, fresh circuits must be used each time the computation is run.

A second regime uses secret-sharing, as exemplified by the GMW protocol [51]. In this regime, both parties participate in the evaluation of the circuit. Each party holds its share of the value of a wire. During circuit evaluation both parties interact via a series of OT steps to exchange their wire shares to execute a gate in the circuit. There is an extensive body of research in both regimes to optimize circuit evaluation, reduce the number of OT rounds, and to enable support for integer and floating point arithmetic and Boolean operations. Although our Pd-Mpc prototype is based on secret-sharing, this section discusses considerations for both regimes (Appendix A1 discusses a GC-based implementation of Pd-Mpc). While garbling schemes aim to minimize interactive communication, the circuit produced by the circuit garbler is often quite large (we quantify this later in the section) and a fresh circuit must be provided to the circuit evaluator each time the circuit is evaluated. In contrast, secret-sharing based approaches minimize upfront network communication but require multiple interactions among the parties during the course of computation.

Algorithm 2 shows DetectFieldOfView adapted to MPC. We now describe the security and performance considerations that went into the design of Algorithm 2.

Security Considerations. Note that Algorithm 1 only considers inputs from *one* drone. To protect citizen privacy, the algorithm needs to be executed with the coordinates of *all* drones in the city. If Algorithm 1 were naïvely iterated over all drones, it would be insecure, as described below.

First, note that the output must not be revealed to the regulatory authority because it could simply use the Boolean result of Algorithm 1 to determine the set of drones in the vicinity of the citizen, and obtain an estimate of the citizen's location. We therefore modify the protocols to only reveal the output to the citizen. In secret-sharing MPC protocols, this goal is accomplished by ensuring that only the regulatory authority reveals its wire shares for the output gates to the citizen, but not vice-versa. In a GC-based setup, one could designate the citizen as the circuit garbler and the regulatory authority as the circuit evaluator. The citizen would modify the garbled tables for the output gates reveal only encoded values (seen by the regulatory authority), rather than the clear-text result of the computation. The encoded values can be decoded by the citizen, but not the regulatory authority.

Second, even if the output were revealed only to the citizen, the nature of the computation in Algorithm 1 has a subtle side-channel when executed on individual drones. While we can expect that there will be a few hundred drones at city-scale (in the near-term, scaling to a few thousand drones in 10-15 years), we only expect a small fraction of these to be within the vicinity radius of the citizen. For example, an Airbus study [15] estimated an average of 16,667 delivery drone flights per hour over Paris (roughly 100km^2) by 2035. This translates to under two drone flights per hour over a fixed 100m^2 area, assuming a uniform distribution. As a result, we can expect that the more complex computation on lines 3 to 6 will be triggered only for a few drones. In secret-sharing schemes, the side-channel manifests as network messages (*i.e.,* the OT steps) exchanged during interactive execution. The regulatory authority can observe the number/size of messages exchanged during circuit

Algorithm 2: DetectFieldOfView in MPC.

```
Input: From citizen: $\Gamma_c$, $\lambda_c$, LatVicinity, LongVicinity, citizen's masking
       factors (drawn from $\mathbb{R}^+$): $c\Gamma_1, c\lambda_1, \ldots, c\Gamma_n, c\lambda_n$.
Input: From regulatory authority, identity, position and $\overrightarrow{D}$ vector for all
       drones: ($\mathrm{Id}^i, \Gamma_t^i, \lambda_t^i, \overrightarrow{\mathbf{D}}^i$) for all drones in city ($i$=1...$n$), regulatory
       authority's masking factors (from $\mathbb{R}^+$): $r\Gamma_1, r\lambda_1, \ldots, r\Gamma_n, r\lambda_n$.
Input: Publicly-known: $\theta_1, \ldots, \theta_n$ for all drones.
Output: Revealed to citizen:
        $\langle \mathrm{DotP}^i, \mathrm{NormSquare}^i, \mathrm{NearbyLat}^i, \mathrm{NearbyLong}^i \rangle$ for all drones.
M1  [$\Gamma$diff] = LatitudeDiff($\Gamma_t^1, \ldots, \Gamma_t^n, \Gamma_c$)
M2  [$\lambda$diff] = LongitudeDiff($\lambda_t^1, \ldots, \lambda_t^n, \lambda_c$)
M3  for $i \leftarrow 1$ to $n$ do
M4      $\mathrm{NearbyLat}^i = (\Gamma\mathrm{diff}[i] - \mathrm{LatVicinity}) \times c\Gamma_i \times r\Gamma_i$
M5      $\mathrm{NearbyLong}^i = (\lambda\mathrm{diff}[i] - \mathrm{LongVicinity}) \times c\lambda_i \times r\lambda_i$
M6      $\overrightarrow{\mathbf{C}}^i$ = Vectorize($\Gamma_t^i, \lambda_t^i, \Gamma_c, \lambda_c$)
M7      $\mathrm{DotP}^i = \overrightarrow{\mathbf{D}^i} \cdot \overrightarrow{\mathbf{C}^i}$
M8      $\mathrm{NormSquare}^i = (|\mathbf{D}^i|)^2 \times (|\mathbf{C}^i|)^2$
M9      Result.add($\langle \mathrm{Id}^i, \mathrm{DotP}^i, \mathrm{NormSquare}^i, \mathrm{NearbyLat}^i, \mathrm{NearbyLong}^i \rangle$)
M10 end
M11 return Result (revealed only to citizen)
    // The steps below happen on the citizen's phone (no MPC)
M12 for $\langle \mathrm{Id}^i, \mathrm{DotP}^i, \mathrm{NormSquare}^i, \mathrm{NearbyLat}^i, \mathrm{NearbyLong}^i \rangle \in$ Result do
M13     if ($\mathrm{NearbyLat}^i \leq 0$) and ($\mathrm{NearbyLong}^i \leq 0$) then
M14         $\phi_i = \arccos(\mathrm{DotP}^i / \sqrt{\mathrm{NormSquare}^i})$
M15         if ($\phi_i \leq \theta_i$) then shortlist the drone $\mathrm{Id}^i$.
M16     end
M17 end
```

evaluation and obtain an estimate of the number of drones in the vicinity of the querying citizen. In turn, this may be used to localize the citizen using the current density of drones in various locations of the city. In a GC-based setup, this side-channel would instead manifest as timing differences: the circuit would take longer to execute for drones in the vicinity of the citizen. If the regulatory authority is the circuit evaluator, this side-channel reveals drones near the citizen.

To mitigate the side-channel, we run the computation in bulk for all drones. That is, rather than running the algorithm for each drone, we modify its inputs so that the regulatory authority provides the coordinates for all drones in one go. The algorithm iterates the distance computation over all the drones. Since all this happens *within the MPC computation* the regulatory authority is oblivious to the identity of the drones that are in the citizen's vicinity.

However, running the computation in bulk for all drones alone is not sufficient for security. If lines 3 to 6 of Algorithm 1 were executed conditionally only on drones in the citizen's vicinity, it exposes a subtle side-channel. In a secret-shared MPC setup, the choice of whether the computation within the conditional is executed (or not) would determine the number of OT messages exchanged between the regulatory authority and the citizen's mobile device. This, in turn, leaks information about the number of drones in the citizen's vicinity, which is undesirable. A similar side-channel would exist in a GC-based setup as well, in which the execution (or not) of the conditional would manifest as timing differences, observable by the regulatory authority, which evaluates the circuit.

As a result, we designed Algorithm 2 to run the steps to compute ϕ for *all* drones, and not just those drones that are in the vicinity of the citizen. By executing the circuit on all drones in the city, Algorithm 2 makes the computation *oblivious* to the number of drones in the citizen's vicinity.

Our choice of using secret-shared MPC over garbled-circuit-based MPC was motivated by the size of the circuits needed to achieve the security properties discussed above. Although Algorithm 2 logically depicts the computation iterating over all the drones using a loop, the circuit represents the computation on lines M4 to M9 with the loop unrolled. Note that this is possible because the value of n is predetermined, allowing the circuit to be generated for fixed values of n. In experiments with both secret-shared and GC-based MPC, we observed that the size of the circuits generated in a GC-based setup were 10-100× larger than the circuit sizes in a secret-shared setup. For example, for n=1000, the size of the circuit in our secret-shared MPC setup (using MOTION [20]) is about 6.5MB. In contrast, the size of the garbled circuit (generated using EMP [95], a popular GC framework) was 685MB. Further, note that in a GC-based setup, a *fresh* circuit must be used each time the two parties engage in computation, thus requiring the exchange of 685MB *each time* the citizen queries the regulatory authority, thus making the entire setup prohibitively expensive.

Performance Optimizations. Our MPC encoding of Algorithm 1 has four key performance optimizations:

① *Distance computation.* Line 1 of Algorithm 1 computes the distance between the drone and the citizen. The Euclidean distance between the drone and the citizen, given their respective GPS coordinates, is computed using Haversine's formula [87]. Computing the Haversine formula involves trigonometric functions and square-root operations, which we wanted to avoid because they are well-known sources of inefficiencies in MPC algorithms [6, 75]. For example, computing distance between a pair of points using Haversine's formula in MOTION [20] requires a circuit with 284 gates and results in 179.9KB network traffic. Vincenty's method [94] is more precise than the Haversine formula, but is even more expensive.

We thus approximate the distance between the citizen and the drone using the difference between the GPS coordinates of the citizen and the drone. A difference of 0.001 degrees in the latitude values of two locations corresponds to a distance of 111 meters between them along the North-South axis [55]. Likewise, the difference between their longitude values estimates their distance along the East-West axis (after suitably normalizing based on the latitude values, to account for an equirectangular projection). Subtraction operations can be implemented cheaply within MPC algorithms. Thus we modify our approach—instead of providing a Euclidean distance as a vicinity threshold, the citizen supplies a threshold of the difference between latitudes and longitudes. In our implementation, approximating the distance using this method requires a circuit with just 6 gates, and one distance computation costs just 0.239KB of network traffic. In Algorithm 2, lines M1-M2 (LatitudeDiff and LongitudeDiff) compute the differences between the citizen's coordinates and the drone's coordinates in bulk, as an n×1 matrix.

② *Lifting Vectorize.* Observe that the call to Vectorize on line 3 in Algorithm 1 only uses inputs from the regulatory authority. This computation therefore does not have to execute as an MPC circuit. The regulatory authority can instead provide the $\overrightarrow{\mathbf{D}}$ vectors of drones as an input to the algorithm.

③ *Early termination.* The computation on line 5 of Algorithm 1 requires inputs from both the citizen ($\overrightarrow{\mathbf{C}}$) and the regulatory authority ($\overrightarrow{\mathbf{D}}$) to compute the angle ϕ. This calculation involves a division, a square root and an arccos operation. The square root appears as the final step in the calculation of the product of the L2-norms $|\overrightarrow{\mathbf{D}}|\times|\overrightarrow{\mathbf{C}}|$. Each of these operations is expensive in MPC.

We therefore modify the algorithm to omit the costly operations. The algorithm computes the value $\overrightarrow{\mathbf{D}} \cdot \overrightarrow{\mathbf{C}}$. It omits the square-root step in the computation of the norms within the MPC algorithm, leaving us with the values $|\overrightarrow{\mathbf{D}}|^2$ and $|\overrightarrow{\mathbf{C}}|^2$ (the square of the L2-norms), which we simply multiply. It reveals these values only to the citizen, who can then proceed with the square-root, division and arccos (in plaintext) on their mobile device to compute ϕ and shortlist drones accordingly. These steps appear as lines M7-M15 of Algorithm 2. This approach results in considerable savings because the generated circuits are much smaller. In turn, this results in much reduced network communication overheads. For example, in our implementation of the computation in lines 5 and 6 (of Algorithm 1), early termination reduces the network traffic for *one* iteration of the loop from 21.5MB to 45.1KB. This is because in a secret-sharing MPC regime, early termination avoids additional OT steps that would otherwise be required if the computation happened within MPC. In a GC-based setup, early termination would reduce the communication from the garbler to the evaluator because of the reduction in the size of the circuit to be garbled.

There are two minor downsides to this approach. The first is that (unlike in Algorithm 1) the citizen learns the identities of all drones in their vicinity, and not just those of the drones that have captured their footage in their field of view. However, we feel that this is an acceptable tradeoff. Some drone-installed tracking devices already broadcast their their presence to nearby devices [86] and the citizen is likely to be aware of the identities of drones close-by.

The second is that the value of θ associated with the drone must be available to the citizen to perform the comparison on line 6 of Algorithm 1. We feel that the values of θ are not sensitive, and can be revealed publicly. In fact, the specification of most commercially-available drones is already available publicly, and the citizen would likely be able to compute the values of θ for most brands of drones. Revealing θ values only provides clues to the citizen about the brand of the delivery drone used, which we again feel is an acceptable tradeoff for performance. However, if delivery drone companies are hesitant to reveal θ values, we note that line 6 alone can be encoded as a separate MPC computation, with the citizen supplying the ϕ values computed by the mobile application.

④ *Avoiding Boolean comparisons in MPC.* Observe that the citizen performs the computations on lines M14 and M15 only on drones that are located within the vicinity threshold. The MPC computation can simply reveal a Boolean value for each drone, informing the citizen whether that drone is in the citizen's vicinity. For drone i in Algorithm 2, this Boolean is $((\Gamma\mathsf{diff}[i]$ - $\mathsf{LatVicinity}) \leq 0) \wedge ((\lambda\mathsf{diff}[i]$ - $\mathsf{LongVicinity}) \leq 0)$. Instead, Algorithm reveals the *sign-preserving masked* values $\mathsf{NearbyLat}^i$ and $\mathsf{NearbyLong}^i$ to the citizen, and the comparison is performed on the citizen's mobile device.

We chose to reveal the values of $\mathsf{NearbyLat}^i$ and $\mathsf{NearbyLong}^i$ to the citizen to utilize optimizations implemented in the MOTION framework [20]. Observe that the MPC computation in Algorithm 2 is purely *arithmetic* in nature (with only +, − and × operators). MOTION uses algorithms tailored to make arithmetic computations fast, and introducing a comparison operator into the computation would require MOTION to switch to a Boolean representation of numbers to perform the computation. We observed that switching from the arithmetic to the Boolean world is an expensive operation in the MOTION framework. For example, with n=1000 drones, we observed that running Algorithm 2 consumes approximately 6.59MB. In contrast, computing the Boolean value $((\Gamma\mathsf{diff}[i]$ - $\mathsf{LatVicinity}) \leq 0) \wedge ((\lambda\mathsf{diff}[i]$ - $\mathsf{LongVicinity}) \leq 0)$ within MPC and revealing only the final result to the citizen consumes approximately 104.7MB.[2]

Number of drones in city	Data consumed (MBs)
100	0.69
200	1.34
500	3.31
1000	6.59
2000	13.14
10,000	65.60

Figure 3: Per-query mobile data usage (MBs) on citizen's phone. The average power draw was ~0.96mAh per MB of data consumed, as measured with the Battery Historian tool [48].

3.3 Implementation

As previously discussed, we used MOTION [20] to implement Algorithm 2. The regulatory authority reveals the value of n in Algorithm 2 (the number of drones), and the circuit is generated for that value of n. Our experimental results indicate that our secret-sharing-based implementation scales to hundreds of drones at city-scale. With a city-scale deployment of a 1000 drones, each query consumes just 6.59MB of mobile data. Assuming the citizen sends queries every 5 minutes, the bandwidth is about half of streaming a low-resolution video on YouTube (see §3.4).

These results already are within the realm of practicality for near-term city-scale deployment of delivery drones. However, the bandwidth consumption and scalability of our approach can be improved further with a number of optimizations. Recent advances show that several steps of the OT can be completed as a pre-computation step, further reducing mobile data consumption [29]. Several MPC frameworks optimize mixed Boolean and arithmetic computations, and allow efficient switching between the Boolean and arithmetic worlds (*e.g.,* [35, 75]), which can also be explored.

3.4 Evaluation of Pd-Mpc

We evaluated Pd-Mpc by studying how it performs under a simulated city-scale drone deployment. To model the regulatory authority server, we used a Intel Core i7-7700 (3.60GHz) CPU with 16GB RAM, running Linux 5.11.0-37 (Ubuntu 20.04). To model the citizen's mobile device, we used a NVidia Xavier NX board with (ARMv8.2 64-bit 6-core CPU, with 8GB RAM), running Linux for Tegra [58]. We chose this platform because there are no publicly-available MPC frameworks for Android devices in the public domain. To our knowledge PAL [65] is the only MPC framework for Android, but is not publicly available and supports an older version of Android.

[2] $\mathsf{NearbyLat}^i$ and $\mathsf{NearbyLong}^i$ are *masked* values of $(\Gamma\mathsf{diff}[i]$ - $\mathsf{LatVicinity})$ and $(\lambda\mathsf{diff}[i]$ - $\mathsf{LongVicinity})$, respectively, with the citizen and the regulatory each providing masking factors $c\Gamma_i$, $c\lambda_i$, $r\Gamma_i$, $r\lambda_i$ as additional inputs to the algorithm. The masking factors (drawn from $\mathbb{R}^+$) serve to protect the raw values of $(\Gamma\mathsf{diff}[i]$ - $\mathsf{LatVicinity})$ and $(\lambda\mathsf{diff}[i]$ - $\mathsf{LongVicinity})$ from both the regulatory authority and the citizen, but *preserve the sign of the result.* Masking is required because the raw values would reveal the locations of the drones to the citizen (or the location of the citizen to the regulatory authority). Note that $\Gamma\mathsf{diff}[i]$ and $\lambda\mathsf{diff}[i]$ are bounded in range as they denote differences of latitudes and longitudes. Thus, the masking factors can be chosen from a large-enough range of enough range of the set of $\mathbb{R}^+$ numbers so that the multiplications on lines M4 and M5 are sign-preserving, *i.e.,* no arithmetic overflows. See Appendix A2 for a detailed treatment of arithmetic overflows.

↓ Number of drones in city	**Query latency at citizen's device**	
	LAN (milliseconds)	**WAN (seconds)**
100	3.73±1.40	4.52±1.29
200	10.77±2.76	8.84±1.59
500	28.36±7.63	13.36±1.96
1000	78.66±15.23	16.33±4.09
2000	153.56±28.29	23.82±4.06
10,000	1281.32±126.20	65.89±29.02

Average RTT between the citizen and regulatory authority on our 1Gbps LAN is 0.220ms (measured with `ping`). In WAN, the citizen is connected via a 4G mobile data network, and the regulatory authority is an instance on Azure US East. The average RTT is 253.28ms, and the citizen's data upload speed during the experiment varied between 1.89Mbps to 12.96Mbps (averaging 6.78Mbps), measured with SpeedTest [89].

Figure 4: Overall query latency at citizen's mobile device.

Data Consumption on Citizen's Mobile Device. We measured the mobile data consumption on the citizen's device to pose queries to the regulatory authority. Figure 3 reports these measurements. The mobile data consumed depends the number of drones deployed in the city, which in turn determines the size of the circuit and therefore the OT messages that must be exchanged (*i.e.,* the value of n in Algorithm 2). Thus, for our experiments we crafted inputs to our MPC implementation that vary the number of drones.

Observe from Figure 3 that mobile data usage increases as the number of drones in the city-wide deployment (n) increases. This is because the size of the circuit increases proportionally with n, thereby resulting in a larger amount of data to be exchanged between the citizen's phone to the regulatory authority. The mobile data usage includes transmitting the circuit itself and OT during the computation. For a deployment with 1000 drones city-wide, each query from the citizen's phone consumes about 6.59MB of mobile data. Assuming that the citizen sends queries every 5 minutes, this translates to approximately 79MB of mobile data consumption an hour. This number is about half the network bandwidth required to stream a low-resolution video from YouTube (135MB/hour at 426x240p resolution, with a bitrate of 300Kbps). We measured an average power draw of 0.96mAh to transmit 1MB of data, which translates to roughly 76mAh per hour at this query frequency, or 1.2% utilization per hour of a standard 6000mAh smartphone battery. While there have been a number of trials of delivery drones, they are yet to be deployed at a large scale, and we expect the growth in this sector to be gradual. Considering, for example, the Airbus study cited earlier [15], which projects 16,667 drones per hour over the city of Paris only by the year 2035, we feel that 1000 drones represents a conservative estimate of a near-term city-scale drone deployment. For a deployment with 10,000 drones city-wide, each query from the citizen's phone consumes about 65.6MB. Further improvements to MPC technology are needed to reduce this number and scale MPC city-wide as drone deployments increase.

Latency at Citizen's Mobile Device. We measured the end-to-end latency observed by a citizen from the time a query is issued to the time that the regulatory authority responds. We studied the latency both when the citizen and regulatory authority are on a 1Gbps LAN, and also on a slow WAN with highly variable upload speeds, in which the citizen's mobile device, connected on a 4G mobile data network (a mobile provider based in India), contacts the regulatory authority that runs on a virtual machine hosted on Azure (US East). Figure 4 reports the results of our experiments (average and standard deviation reported over 5 runs), which show that even in a slow WAN setting with n=1000, the citizen can obtain query results in approximately 16±4 seconds.

Although our evaluation is citizen-centric, note that it also provides insight into the performance of the regulatory authority's server. Even in the most computationally- and communication-heavy case that we considered (10,000 drones city wide), the overall client latency was about 1.28s in a LAN. The regulatory authority can parallelize MPC computation for different queries, therefore easily scaling up to an arbitrary number of querying citizens.

Accuracy of the DetectFieldOfView Algorithm. Finally, we evaluated the precision of our approach at detecting whether an object is in a camera's field of view. For this evaluation, we conducted a field study in which we simulated the setup in Figure 2 using a fixed camera, *i.e.,* we placed the the camera, denoting the drone camera, at specified location with a fixed orientation (*i.e.,* the values of Γ_t, λ_t and the $\overrightarrow{\mathbf{D}}$ vector are known). We set up ground markers denoting the camera's cone of view, *i.e.,* $\pm\theta$ w.r.t. $\overrightarrow{\mathbf{D}}$.

We then repeated an experiment in which placed a person of interest, denoting the citizen, at a specified location (*i.e.,* Γ_c, λ_c is known). Out of 20 trials of this experiment, we placed the person of interest within the camera's field of view in 10 trials, and outside the field of view in 10. DetectFieldOfView precisely determined that the person was within the field of view (or not) in 19 out of the 20 trials. The lone false positive was a case in which the person was close to the boundary of (and within) the field of vision, but was identified as being outside. We attribute this error to the quality of GPS values that we were able to obtain to determine location coordinates (we used the person's Samsung M31 device to determine their GPS coordinates). When we instead simulated the same experiment with Google Maps, using markers to identify the locations of the citizen, drone camera and the drone's direction of motion, DetectFieldOfView was 100% accurate.

4 AUDITING COMPLIANCE WITH Pd-Ros

With the shortlist of drones in hand, the citizen uses Pd-Ros to obtain audit trails from drones and ensure that they are privacy-compliant. Pd-Ros is a set of tools that helps well-intentioned drones achieve the following goals:

- (G1) *Use only sanitized data.* Pd-Ros must ensure that camera data is never used unsanitized.
- (G2) *Offer compliance proof.* Pd-Ros must be able to convince a citizen that goal G1 is satisfied.

To reliably offer proofs of compliance, Pd-Ros assumes the presence of trusted hardware on the drone. This hardware must be capable of performing basic cryptographic operations, offer the ability to securely store an audit trail, and respond to citizen requests with a digitally-signed audit trail. Trusted hardware is typically endowed with a public/private key pair with the private key stored securely in the hardware, and the public key digitally certified by a certificate authority. This public key serves as the drone's identifier.

In our prototype, we use the ARM TrustZone [10], which provides these features.[3] An ARM TrustZone processor can be in one of two *worlds* of execution at any given instant—a *secure world* or a

[3] While ARM TrustZone has several known vulnerabilities [23], our use of the TrustZone in this paper is merely illustrative. Pd-Ros can be built atop any "secure" TEE with similar features. Side channel attacks (*e.g.,* [78]) against ARM TrustZone TEEs have also been documented in the literature; these are out of scope.

normal world. The secure world, also called the trusted-execution environment (or the TEE), runs trusted software services. The normal world, or the rich-execution environment, runs untrusted applications and is normally the environment in which end-users of the device conduct the bulk of their activities. These features are implemented with traditional hardware-level protection, and a small, trusted, *secure monitor* that executes at a higher processor privilege level than the OS in the secure and normal worlds. The two worlds interact via a secure monitor call (`smc` instruction) that allows world switches. The secure world implements features such as normal world attestation (*e.g.,* as in Knox [14]). ARM TrustZone provides memory isolation for the secure world as a default feature, *i.e.,* the normal world cannot map or access secure world memory. This feature provides us a path to achieve goal **G2** because audit trails and attestation reports can be safely stored in the secure world.

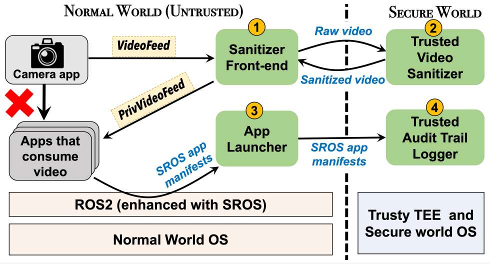
4.1 Background on ROS2

ROS2 is a popular middleware platform for robotics [83]. It is a set of libraries and user-space utilities that provide support for easy development of distributed robotics applications across a federation of robots. ROS2 offers a publish/subscribe model for robotics applications to communicate with each other. Applications publish messages labeled with specific topic names. Applications that subscribe to that topic can receive these messages from the publishing application. ROS2 uses the Data Distribution Service [33, 56] to match pairs of applications that have such a publish/subscribe relationship. Each ROS2 application executes as a process atop the underlying OS, and ROS2 sets up either socket-based or shared-memory communication between a pair of applications.

ROS2 itself does not authenticate message senders, and all messages between applications are transmitted in the clear. This leads to a number of spoofing and eavesdropping attacks (*e.g.,* [38, 39, 63, 81]). Moreover, any application can publish or subscribe to any topic. This leads to situations where a malicious application can publish a video feed under the same topic as a genuine application, and confuse downstream applications that consume the video feed. The community has therefore developed Secure ROS (SROS) to overcome these shortcomings [97–99]. In SROS, TLS is used to secure the communication between applications. Further, each application must provide a manifest that declares the list of topics to which that application publishes or subscribes. The application's code and this manifest are then cryptographically bound via an X.509 certificate, signed by a trusted third party. SROS uses the X.509 certificate to detect and prevent the launch of any applications whose code or manifest have been modified. SROS also ensures that an application can only publish or subscribe to the topics explicitly identified in its manifest. Pd-Ros builds upon ROS2 extended with SROS (we will use "ROS2" to refer to ROS2 enhanced with SROS).

4.2 Core Components of Pd-Ros

On ARM TrustZone-enabled delivery drones, ROS2 and the corresponding application ecosystem execute in the normal world. The secure world runs a minimal trusted OS and trusted applications (the Trusty TEE [52] in our Pd-Ros prototype). Pd-Ros offers tools implemented in the normal world and in the secure world to restrict access to the raw video feed on drones in which the camera cannot



Pd-Ros provides well-intentioned drones tools to ensure that video footage is suitably sanitized, and to provide an audit trail to citizens to convince them so. Pd-Ros introduces: ① A ROS2 application that subscribes to the camera's feed (topic *VideoFeed*). On a well-intentioned drone, no other applications will subscribe to this topic (they subscribe to *PrivVideoFeed*); ② A trusted video sanitizer that runs in the secure world and applies region-specific privacy policies to the video feeds; ③ An agent in the normal world that collects the SROS manifests of applications that are launched for execution; ④ An audit logger that stores SROS manifests, and digitally signs and sends them in response to citizen queries.

Figure 5: Setup of ARM TrustZone-based drone with Pd-Ros.

be exclusively assigned to the secure world. It requires that raw access to camera hardware be restricted to a single ROS2 application. This requirement can easily be enforced by the normal world OS.

Suppose that this camera application publishes its feed to a topic called *VideoFeed*. Downstream applications can consume the video feed by subscribing to this topic. Pd-Ros requires that applications on well-intentioned drones subscribe instead to a topic called *PrivVideoFeed*. Pd-Ros provides a dedicated ROS2 application—the Sanitizer Front-End (Sanitizer-FE)—that exclusively subscribes to *VideoFeed* and publishes to *PrivVideoFeed* (see Figure 5). Sanitizer-FE uses the traditional publish/subscribe abstraction, thereby allowing other downstream ROS2 applications to interact with it without requiring any invasive changes to their code. At the back-end, Sanitizer-FE uses a secure monitor call (`smc`) to perform a world switch, and interact with a trusted video sanitizer in the secure world. This video sanitizer is entrusted with applying the region-specific privacy policy, *e.g.*, blurring faces in the video feed. ROS2 applications that require raw access to *VideoFeed* can be accommodated as exceptional cases after vetting that they do not intentionally leak the feed.

This setup suffices to ensure goal **G1** on a well-intentioned drone. However, the drone must also convince a citizen that this setup exists on the drone (goal **G2**). To accomplish goal **G2**, Pd-Ros relies on the way application manifests work in SROS. As previously discussed, the manifest is cryptographically-bound to the application's identity, and SROS uses the X.509 certificate of the application to check that the manifest and the application's code have not been tampered. The manifest specifies the topics that the application publishes/subscribes to, and SROS ensures that the application does not deviate from this specification at runtime.

Pd-Ros extends ROS2's application launcher to store an application's manifest in the secure world when it is launched in the normal world. The trusted audit logger in the secure world attests the normal world kernel, ROS2 and SROS and stores the attestation report in the audit trail whenever an application is launched. Entries in the audit trail are time-stamped, and the citizen either requests the entire audit trail, or a snippet for a particular time interval. Upon a citizen query, the secure world digitally signs and sends the audit trail. The citizen then uses the audit trail as follows:

① *Check normal world OS/ROS2/SROS integrity.* The citizen first uses the attestation report to verify the integrity of the normal world OS, ROS2, and SROS. This step is critical because Pd-Ros relies on the normal world OS to ensure that access to the camera hardware is restricted to a single application. Because Pd-Ros relies on SROS to enforce application manifests, the citizen must ensure the integrity of ROS2/SROS in the normal world.

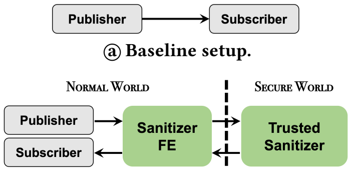
② *Check integrity of Sanitizer-FE and ROS2 application launcher.* As is standard [85], we assume that the normal world OS also includes in the attestation report the integrity measurements of applications that it launches. The citizen uses these measurements to verify the integrity of Sanitizer-FE and the application launcher.

③ *Check publish/subscribe patterns.* SROS application manifests limit how applications communicate. Thus, the citizen simply needs to verify that applications that execute in the normal world do not subscribe to any topics published by the camera application (*i.e., VideoFeed*), and that Sanitizer-FE subscribes only to *VideoFeed* and publishes to a single topic, *PrivVideoFeed*. Together with the runtime assurance provided by SROS and the trusted video sanitizer in the secure world, the citizen can be assured that the video footage is being sanitized before use. While the topics specified in the manifest of an application represent the sandbox within which the application can operate, they may not always publish or subscribe to all the topics in the manifest. Although not in our prototype, Pd-Ros could use an agent to dynamically query ROS2 to determine the publish/subscribe graph, and store that instead in the audit trail.

Citizens can either query the shortlisted drones directly (if such communication support exists), or via the regulatory authority. Such queries must obviously be anonymous to protect the citizen's identity, failing which the regulatory authority can simply use the set of queries from a citizen to compromise their location privacy. Fortunately, such a query interface can easily implemented using well-known methods (*e.g.,* Tor).

The assurances provided by the audit trail rely on certain assumptions that are standard in hardware-based attestation. For example, we cannot defend against zero-day exploits on SROS, ROS2, the normal world kernel, or any of the components of Pd-Ros. We also assume that SROS completely mediates application communication within the normal world. A malicious ROS2 application can attempt to bypass SROS by directly invoking OS abstractions, *e.g.,* via raw sockets, to communicate with a colluding application [16]. Prior work has attempted to harden the normal world kernel against zero-day exploits [14, 50], and to ensure that applications only communicate under the purview of SROS [9, 16]. Those methods also apply to our setting. §A3 provides a detailed analysis.

Given our goals, it is natural to ask whether audit trails produced by Pd-Ros alone do not suffice to protect citizen privacy. The regulatory authority could itself periodically query all delivery drones in operation, check the audit trails to determine whether the drones are in compliance, and publish a compliance report for all drones on a public forum for all citizens to see. If the compliance report also includes the locations of drones (provided a drone operator permits the release of the location information for all its drones, city-wide), citizens can determine the compliance status of drones in their vicinity using this public forum.



ⓐ **Baseline setup.**

ⓑ **Setup with redirection and blurring.**

Metric	**Baseline**	**With redirection**	
Time	16.643s	16.687s	(+0.26%)
CPU Utilization	18.68%	31.82%	(+70.34%)
Power Use:			
• *System 5V rail*	4138.58mW	4980.73mW	(+20.34%)
• *SoC rail*	1217.15mW	1250.11mW	(+2.71%)
• *CPU+GPU rail*	744.22mW	1502.87mW	(+101.8%)

ⓒ **Performance impact of flow redirection on NVidia Xavier.**

Figure 6: Pd-Ros experimental setup and results.

The key shortcoming with this deployment model is that does not provide any mechanism for the citizens to seek further accountability that is otherwise possible with Pd-Mpc. For example, with Pd-Mpc, the citizens know precisely which drones have captured their footage. This allows the possibility of an accountability mechanism via which citizens can request their footage from those drones and verify that it has been sanitized. This is akin to citizens being able to request their activity reports on platforms such as Google or Facebook, or citizens being able to view their footage on Google Street View and confirm that it has been sanitized.

4.3 Evaluation of Pd-Ros

We implemented and evaluated Pd-Ros on a Nvidia Xavier NX development kit, equipped with a 6-core Nvidia Carmel ARMv8.2 64-bit CPU and 8GB RAM. It also has a 384 core GPU with 48 tensor cores that makes it ideal for heavy video processing. This hardware is similar to that found as the companion boards on many commercial drones. We chose this development kit because it is equipped with the TrustZone and has an open and programmable secure world. We configured the secure world to run the Trusty TEE [52], which allows for easy development of trusted applications. The normal world runs Linux for Tegra (L4T) [58], which consists of the Linux kernel 4.9, a bootloader, and supporting drivers. We installed a ROS2-based environment in the normal world.

To evaluate the cost of sanitizing video feeds, we consider the setup shown in Figure 6. We built a pair of ROS2 applications, the first of which publishes a video feed that the second consumes directly, as shown in Figure 6ⓐ. This is our baseline, which we run with a workload in which the publisher sends a video feed consisting of 250 frames (resolution 320×240 pixels, at 30 frames/second). The publisher converts each frame into ROS2's message format, which the subscriber receives and converts back to a video frame. We then implemented redirection and sanitization of the video feed as shown in Figure 6ⓑ, in which Sanitizer-FE exclusively subscribes to the feed of the publisher, and redirects the flow to a trusted sanitizer in the secure world. The trusted sanitizer applies a simple box blur filter to the image. We intentionally kept the functionality of the trusted sanitizer simple to show the basic cost of redirection. While prior user studies in a related domain have shown that even a simple blur filter can improve the perception of privacy for bystander citizens [40], recent work has shown that some facial features can still be identified with blurring [105]. More complex image processing logic can obviously be built within the trusted sanitizer, with a corresponding increase in resource consumption. The trusted sanitizer returns the modified feed to Sanitizer-FE, from which the subscriber application consumes it.

Figure 6ⓒ presents the results of our experiments. We measured the end-to-end latency of the application workflow (from publishing to receiving the video feed), the CPU utilization. We also used the in-built 3-channel INA3221 power monitor on the Xavier board to measure power consumption at the system, SoC and CPU+GPU rails, respectively. As our results show, the overall application latency on this lightly-loaded drone is minimally impacted, with the video being blurred in near real time. This operation only led to a modest increase in CPU utilization and power draw.

We also evaluated the overall cost of attesting the normal world (kernel, ROS2, and SROS), and storing application manifests in the secure world. Recall this step is performed when the application is launched, and thus increases launch time. We measured the time to launch the `cam2image` application, which is part of the standard ROS2 distribution. Without Pd-Ros this application takes 4.78s to launch on our hardware platform, while it takes 10.66s with Pd-Ros.

5 LIMITATIONS AND IMPROVEMENTS

Our goal in developing Privadome was to explore whether an end-to-end citizen-centric privacy protection solution is feasible in the presence of delivery drones, and to provide rigorous assurances (using trusted hardware) to the citizen that their data has been sanitized. However, we acknowledge that Privadome needs to be improved in a number of ways for practical deployment. In this section, we list a few possibilities for improvement.

Scaling Privadome to Larger Number of Drones. When designing Pd-Mpc, we were guided by the results of the Airbus study [15], which predicts an hourly traffic density of roughly 17,000 drones over a metropolitan city the size of Paris by the year 2035. As our results with Pd-Mpc showed, there is a sharp increase in both query latency and bandwidth when we assume a drone density of over 1000 drones at a given time instant. While we believe that Privadome will suffice for near-term deployments of delivery drones, further research is needed to scale it up to the deployment densities predicted for the long term by the Airbus study. The primary bottlenecks to scaling Pd-Mpc, in particular, are the latency and bandwidth requirements of MPC. There are a number of possibilities that can be explored to improve these aspects of Pd-Mpc:

- *Leveraging cryptographic advances.* Our Pd-Mpc prototype discussed in §3 is based on secret-sharing, which is an interactive protocol that requires multiple rounds of oblivious transfer (OT). Our prototype currently uses the OT algorithms implemented in MOTION without modifications. However, recent advances promise up to 1300× reduction in the amount of live network traffic needed in the OT step [29] via pre-computation, and these advances can potentially be explored to scale up Pd-Mpc.
- *Offline querying.* As presented, Pd-Mpc sends live privacy queries from the citizen's mobile device thus allowing the citizen to obtain an instant assessment of how their privacy is impacted. However, with some support from the regulatory authority, Pd-Mpc can also be adapted to be used as an offline auditing system. If the regulatory authority maintains a historic log of delivery drone locations, then MPC queries sent by citizens can be executed against this log. An agent on the citizen's mobile device can keep a history of the device's location, and periodically send batched privacy queries using the set of locations that the citizen has visited. The regulatory authority reconstructs the snapshot of drone locations (using the historic log), and executes the MPC protocol against the client device's location at a given instant in time to perform the computation discussed in §3.1. In fact, with this approach, the client's mobile device need not be involved in the MPC computation at all. The location history can be offloaded from the mobile device to a more well-equipped edge-computing device (or cloudlet) that the citizen trusts, thereby further relieving the mobile device from having to participate in the MPC protocol. In turn, this reduces the bandwidth requirements on the mobile device to that of simply transferring the location history to the edge-computing device. The design of Pd-Ros already lends itself to an offline auditing system, thereby converting the entire Privadome system into one that places low bandwidth and energy requirements on the client's mobile device.

Safeguards for Delivery Drone Operators. Privadome's design assumes that delivery drone operators will share the locations of their delivery drones with regulatory authorities. This assumption is justified by the requirements imposed by various countries on delivery drone operators that requires them to share the live location of their drones [13, 34, 47, 54, 72, 73, 101]. However, delivery drone operators will reasonably want the regulatory authority keep this data private from citizens and their competitors.

While we built Privadome as a citizen-privacy-centric system, its approach (particularly Algorithm 2) does reveal the identity of nearby drones to a participating citizen in response to a privacy query. This step was essential because the identity of drones was used subsequently by Pd-Ros to contact the corresponding drones and obtain an audit trail of privacy compliance. However, this approach has the unintentional side-effect of allowing a group of colluding citizens in the city to crowd-source a map of delivery drone locations in the city. Each citizen simply publishes the identity of the drones in their vicinity to a shared Web site, using their own location as an approximate location for a particular drone's identity. With sufficiently many citizens contributing, this can possibly lead to a real-time map of drones in the city.

Again, it is possible to mitigate this threat with a few changes to the algorithm, and the cooperation of the regulatory authority. This approach requires the regulatory authority to establish a server trusted both by citizens and drone delivery authorities that will generate proxy identifiers (*e.g.,* a random 1024-bit integer) for each drone. The MPC algorithm 2 is then modified, so that instead of revealing the raw identities of drones in the vicinity of the citizen, it reveals a proxy identifier for each drone. Even though queries in Pd-Mpc are anonymous, the algorithm must use fresh proxies for the same drone across queries to prevent correlation of drone locations across queries. Thus, in this approach, each drone may be associated with multiple proxy identifiers on the trusted server. The raw identities of the drones are thus never revealed to the citizen. Pd-Ros will also have to be suitably modified to route queries to drones via the trusted server. This server will in turn obtain attestation reports with the individual delivery drone, and report the results back to the citizen.

6 RELATED WORK

A number of prior works have considered security and privacy of drones and other aerial vehicles, broadly considered. Nassi *et al.* [69] provide a comprehensive overview of the security and privacy issues in the era of drones. Recent work has attempted to detect privacy violations committed by drones that are controlled by a ground-based operator equipped with a first-person view (FPV). These works leverage the observation that such a drone must wirelessly communicate with the operator to export the camera's view to the FPV. Wi-Fly [18] aims to detect drones that hover outside the windows of homes. It uses a window-mounted sensor that detects a drone approaching the window by studying variations in the received signal strength (RSS) at the sensor. Nassi *et al.*'s work [17, 67, 68] detects whether an object (or person) is captured in the FPV. Their work is based on the observation that if an object is being recorded in the FPV, then physical perturbations of the object (*e.g.,* shining light on it) manifest as observable changes in the encrypted wireless channel between the drone and the remote control. They use a ground-based detector to intercept this encrypted wireless channel, and then employ cryptanalysis techniques that leverage this observation to detect privacy-violating drones.

However, this prior work suffers from three important shortcomings. First, the detection methods are *tailored to drones that export an FPV* to a ground-based operator. They rely in a key way on the detector having access to the wireless channel that the drone uses to export the FPV. They are thus not applicable to autonomous drones, which may not export such an FPV or have a ground-based operator. Second, they *do not offer an end-to-end solution* to a citizen who may wish to determine whether their privacy is violated. That is, while they may help detect that a citizen is captured in the FPV, they offer the citizen no way to deal with the violation or communicate with the drone to query how the recorded video feed is used or stored. Indeed, these methods are not in any way tied to any regulatory framework that offers the citizen to reason about how the captured data is used. Finally, these methods require *active citizen participation*, either by installing detectors on their home windows (in the case of Wi-Fly [18]) or physically perturbing the object/citizen suspected of being observed (in the case of Nassi *et al.*'s work [17, 67, 68]). However, some synergies with Privadome exist. A citizen can use Nassi *et al.*'s methods to detect FPV-based drones that have him/her in their field of view (instead of PD-MPC), and then use PD-ROS to engage with the drone to obtain an audit trail. While Privadome is restricted to delivery drone operations that are overseen by a regulatory authority, Wi-Fly [18] and Nassi *et al.*'s methods [17, 67, 68] apply to any FPV-based drones.

While Wi-Fly and Nassi *et al.*'s work focuses on how individual citizens are impacted, Privaros [16] and AliDrone [61] develop methods to regulate delivery drones over well-demarcated host airspaces that may dictate that specific policies are to be obeyed within the airspace. For example, a college campus or an apartment complex may require the delivery drone to ensure that images and video recorded in the their airspace be blurred suitably, or that drones follow certain pre-identified drone lanes during their visits. Privaros develops mandatory access control extensions for ROS-based drones that can accept and enforce policies specified by ground-based hosts. Like PD-ROS, Privaros also relies on trusted hardware to prove to hosts that the delivery drones are in compliance. However, unlike Privadome, Privaros does not focus on privacy of individual citizens, nor does it offer a method to determine whether a citizen's footage is captured by the drone. AliDrone [61] is tailored to ensure that drones remain on drone lanes during their delivery runes. It uses trusted hardware to securely store proofs-of-alibi (GPS coordinates of the flight path) that can be used to prove to host airspaces that the drone was in compliance. PROTC [60] also uses trusted hardware on drones with a focus on protecting the drone software stack and peripherals from malicious attacks and rootkits [60].

Ding *et al.* [41] consider how location broadcast systems like Remote ID can potentially compromise citizen privacy. They consider attacks whereby an attacker observing the drone's location can correlate drone trajectories with ground-based citizens and their purchases. They also develop methods to quantify privacy risks and generate routes that offer privacy. This work is complementary to Privadome, which is citizen-centric.

In contrast to the above works, which mainly focus on privacy, there is a significant body of work on drone security. Given the near-daily news stores about drones being used to conduct various terrorist attacks (*e.g.,* Venezuela [57], Iraq [32] and Japan [19] being prominent examples), it is not surprising that much of the focus is on detecting unauthorized drones. Methods to detect drones range from the use of radar [45], Lidar [1, 26, 92] radio-frequency [70], computer vision [84], and acoustic signatures [21, 22]. These methods can complement Privadome in city-scale deployments to detect or deter drone flights outside of the purview of the regulatory authority. Security research focused on delivery drones has mainly considered reliable package delivery. Here, prior work has focused on methods to mutually authenticate delivery drones and intended recipients, for example, using the sound signature of the drone [79]. Researchers have also developed methods to ensure that delivery drones are not sabotaged in-flight, by developing methods to detect and avoid projectiles thrown at them [49].

Abidi *et al.* [3] consider a setting in which citizens query aggregate pollution statistics collected from sensors fitted on taxi fleets. As in Privadome, they protect citizen privacy using MPC. However, they also consider the dual problem of protecting the privacy of the taxi fleet using differential privacy, *e.g.,* to hide the distribution of taxis of one fleet operator from a competitor. While Abidi *et al.*'s work is in a different domain, Privadome can borrow similar ideas to also protect drone fleet privacy from competing fleet operators.

7 CONCLUSION

Even as e-commerce companies work on the technology and infrastructure to enable drone-based delivery, citizens are concerned about on how their privacy will be impacted once it sees wide deployment. Clearly, much needs to be done to abate these concerns, in the form of governmental regulation and enforcement methods. Privadome is a step in that direction. Privadome can be integrated with city-scale regulatory authorities that oversee drone operations. It allows citizens equipped with a mobile phone to determine if their footage is recorded by drones in their vicinity, and request an audit trail from those drones to determine if they are privacy compliant. Privadome accomplishes all these goals without revealing the citizen's location, and while only consuming mobile data comparable to streaming low-resolution videos.

Acknowledgments

We gratefully acknowledge financial support from the C3iHub, IIT-Kanpur (via DST, Govt. of India). We also thank the PETS 2024 reviewers for their thoughtful feedback on this work.

REFERENCES

[1] 3DEO. Rogue drone detection and mitigation. https://3deo.biz/applications/drone-detection-and-mitigation.

[2] T. Abera, N. Asokan, L. Davi, J. Ekberg, T. Nyman, A. Paverd, A-R. Sadeghi, and G. Tsudik. C-FLAT: Control-Flow attestation for embedded systems software. In *ACM Conference on Computer and Communications Security*, 2016.

[3] I. Abidi, I. Nangia, P. Aditya, and R. Sen. Privacy in Urban Sensing with Instrumented Fleets, Using Air Pollution Monitoring As A Usecase. In *Network and Distributed Systems Security Symposium*, 2022.

[4] P. Aditya, R. Sen, P. Druschel, S-J. Oh, R. Benenson, M. Fritz, B. Schiele, B. Bhattacharjee, and T. T. Wu. I-Pic: A Platform for Privacy-Compliant Image Capture. In *ACM Conference on Mobile Systems, Applications, and Services*, 2016.

[5] Amazon Prime Air, December 2016. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

[6] A. Aly and N. P. Smart. Benchmarking privacy preserving scientific operations. In *Cryptology ePrint Archive: Report 2019/354*, April 2019.

[7] Amazon.com. Drone Delivery FAQs. https://www.amazon.com/gp/help/customer/display.html?nodeId=T3jxhuvPfQ629BOIL4.

[8] Amazon.com. Drone Delivery FAQs—May 8, 2023 snapshot of reference [7] from the Internet Archive. Archived link: https://web.archive.org/web/20230508231429/https://www.amazon.com/gp/help/customer/display.html?nodeId=T3jxhuvPfQ629BOIL4.

[9] AppArmor and ROS. http://wiki.ros.org/SROS/Tutorials/AppArmorAndROS.

[10] ARM. Security technology building a secure system using TrustZone technology (white paper). *ARM Limited*, 2009. https://community.arm.com/cfs-file/__key/telligent-evolution-components-attachments/01-2057-00-00-00-00-53-99/PRD29_2D00_GENC_2D00_009492C_5F00_trustzone_5F00_security_5F00_whitepaper.pdf.

[11] PrimeCell® Infrastructure AMBA™ 3 TrustZone Protection Controller (BP147)—Revision: r0p0—Technical Overview, November 2004.

[12] United Kingdom-Civil Aviation Authority. The Drone and Model Aircraft Code—Protecting people's privacy. https://register-drones.caa.co.uk/drone-code/protecting-peoples-privacy.

[13] United States Federal Aviation Authority. 86 FR 4390: RemoteID: Remote Identification of Unmanned Aircraft, 15th January 2021. https://www.federalregister.gov/documents/2021/01/15/2020-28948/remote-identification-of-unmanned-aircraft.

[14] A. Azab, P. Ning, J. Shah, Q. Chen, R. Bhutkar, G. Ganesh, J. Ma, and W. Shen. Hypervision across worlds: Real-time kernel protection from the ARM TrustZone secure world. In *ACM Conference on Computer and Communications Security*, 2014.

[15] K. Balakrishnan, J. Polastre, J. Mooberry, R. Golding, and P. Sachs. Blueprint for the Sky—The roadmap for the safe integration of autonomous aircraft, 2018. https://www.airbusutm.com/uam-resources-airbus-blueprint.

[16] R. Beck, A. Vijeev, and V. Ganapathy. Privaros: A Framework for Privacy-Compliant Delivery Drones. In *ACM Conference on Computer and Communications Security*, 2020.

[17] R. Ben-Netanel, B. Nassi, A. Shamir, and Y. Elovici. Detecting Spying Drones. *IEEE Security and Privacy*, 19(1), 2021.

[18] S. Birnbach, R. Baker, and I. Martinovic. Wi-fly?: Detecting privacy invasion attacks by consumer drones. In *Network and Distributed Systems Security Symposium*, 2017.

[19] D. Bolton. Man arrested for landing 'radioactive' drone on Japanese Prime Minister's roof. In *The Independent*, April 24 2015.

[20] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko. MOTION: A Framework for Mixed-Protocol Multi-Party Computation. *ACM Transactions on Privacy and Security*, 8:8:1–8:35, May 2022.

[21] J. Busset, F. Perrodin, P. Wellig, B. Ott, K. Heutschi, T. Rühl, and T. Nussbaumer. Detection and tracking of drones using advanced acoustic cameras. *Unmanned/Unattended Sensors and Sensor Networks XI and Advanced Free-Space Optical Communication Techniques and Applications*, 2015.

[22] E. E. Case, A. M. Zelnio, and B. D. Rigling. Low-cost acoustic array for small UAV detection and tracking. In *IEEE National Aerospace & Electronics Conference*, 2008.

[23] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto. SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems. In *IEEE Symposium on Security and Privacy*, 2020.

[24] V. Chang, P. Chundury, and M. Chetty. "Spiders in the sky": User perceptions of drones, privacy, and security. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2017.

[25] Mike Cherney. Some Want Delivery Drones to Buzz Off. Would Stricter Rules Change Their Minds? *Wall Street Journal*, 2nd August 2019.

[26] P. Church, C. Grebe, J. Matheson, and B. Owens. Aerial and surface security applications using LIDAR. In *Laser Radar Technology and Applications XXIII—International Society for Optics and Photonics*, volume 10636, 2018.

[27] Cisco. Cisco 2022 Data Privacy Benchmark Study, 2022. https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-privacy-benchmark-study-2022.pdf.

[28] CNBC. Amazon says this business opportunity could make you up to $300k a year—here's how to get into the program, September 2018. https://www.cnbc.com/2018/09/06/amazon-delivery-service-partner-program-gets-thousands-of-applications.html.

[29] G. Couteau, P. Rindal, and S. Raghuraman. Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes. In *CRYPTO – Advances in Cryptology*, 2021.

[30] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad. A context-aware security architecture for emerging applications. In *Annual Computer Security Applications Conference*, 2002.

[31] Taylor Covington. Could Delivery Drones Be the Next Tech Privacy Violation? 88% of Americans Think So. *The Zebra*, April 2020. https://www.thezebra.com/resources/home/delivery-drones-survey/.

[32] J. Davison and A. Rasheed. Iraqi PM safe after drone attack on residence, military says. In *Reuters*, November 7 2021.

[33] Data Distribution Service (DDS). https://www.omg.org/spec/DDS/1.4/PDF.

[34] Weibe de Jager. Switzerland launches world's first Remote ID network for drones, September 2021. https://www.dronewatch.eu/switzerland-launches-worlds-first-remote-id-network-for-drones/.

[35] D. Demmler, T. Schneider, and M. Zohner. ABY–A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *Network and Distributed Systems Security Symposium*, 2015.

[36] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7), July 1977.

[37] J. Diaz. Amazon, TikTok, Facebook, others ordered to explain what they do with user data. *National Public Radio*, December 2020. https://www.npr.org/2020/12/15/946583479/amazon-tiktok-facebook-others-ordered-to-explain-what-they-do-with-user-data.

[38] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner. Security for the Robot Operating System. *Robotics and Autonomous Systems*, 98, 2017.

[39] B. Dieber, S. Kacianka, S. Rass, and P. Schartner. Application-level security for ROS-based applications. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016.

[40] M. Dimiccoli, J. Marin, and E. Thomasz. Mitigating bystander privacy concerns in egocentric activity recognition with deep learning and intentional image degradation. *Proceedings of the ACM on Interactive Mobile and Wearable Ubiquitous Technologies*, (132), 2017.

[41] G. Ding, A. Berke, K. H. Degue, H. Balakrishnan, K. Gopalakrishnan, and M. Z. Li. Routing with privacy for drone package delivery systems. In *International Conference on Research in Air Transportation*, 2022.

[42] Drones4Sec – European Federation – General Secretariat by ZENON7, June 2021. https://www.drones4sec.eu/.

[43] How To Get Flipkart Franchise. https://www.steptowardbusiness.com/flipkart-franchise.

[44] Clare Duffy. Google agrees to pay $13 million in Street View privacy case, 2019. CNN Business, July 25, 2019, https://edition.cnn.com/2019/07/22/tech/google-street-view-privacy-lawsuit-settlement/index.html.

[45] T. Eshel. Mobile radar optimized to detect UAVs, precision guided weapons. *Defense Update*, 2013.

[46] ETtech. Swiggy to pilot drone-based deliveries for its grocery service Instamart. *Economic Times*, April 2022. https://m.economictimes.com/tech/startups/swiggy-to-pilot-drone-based-deliveries-for-its-grocery-service-instamart/amp_articleshow/91188144.cms.

[47] Commission Implementing Regulation (EU). Rules and procedures for the operation of unmanned aircraft, May 2019. https://ec.europa.eu/info/law/better-regulation/have-your-say/initiatives/1642-Detailed-rules-on-unmanned-aircrafts_en.

[48] Android for Developers. Profile battery usage with Batterystats and Battery Historian. https://developer.android.com/topic/performance/power/setup-battery-historian.

[49] N. Garg and N. Roy. Acoustic sensing for detecting projectile attacks on small drones. In *HotMobile'20: 21st International Workshop on Mobile Computing Systems and Applications*, 2020.

[50] X. Ge, H. Vijayakumar, and T. Jaeger. Sprobes: Enforcing Kernel Code Integrity on the TrustZone. In *IEEE Workshop on Mobile Security Technologies*, 2014.

[51] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game, or A Completeness Theorem for Protocols with Honest Majority. In *ACM Symposium on the Theory of Computing*, 1987.

[52] Google. Trusty TEE – Android Open Source Project. https://source.android.com/security/trusty.

[53] Government of India. Digital Sky Portal — Office of the Director General of Civil Aviation. https://digitalsky.dgca.gov.in/.

[54] Government of India. Office of the Director General of Civil Aviation: DGCA RPAS Guidance Manual, Revision One of First Edition, June 2019. https://diceindia.org.in/wp-content/uploads/Updated-DGCA-RPAS-Guidance-Manual.pdf.

[55] Approximate Metric Equivalents for Degrees, Minutes, and Seconds, August 2019. https://www.usna.edu/Users/oceano/pguth/md_help/html/approx_equivalents.htm.

[56] Object Management Group. About the Data Distribution Service Specification Version 1.4. https://www.omg.org/spec/DDS/About-DDS/.

[57] C. Koettl and B. Marcolini. A closer look at the drone attack on Maduro in Venezuela, August 2018. https://www.nytimes.com/2018/08/10/world/americas/venezuela-video-analysis.html.

[58] NVIDIA Jetson Linux Driver Package (L4T). https://developer.nvidia.com/embedded/linux-tegra.

[59] M. Lentz, R. Sen, P. Druschel, and B. Bhattacharjee. SeCloak: ARM Trustzone-based Mobile Peripheral Control. In *ACM Symposium on Mobile Systems, Applications, and Services*, 2018.

[60] R. Liu and M. Srivastava. PROTC: Protecting drone's peripherals through ARM TrustZone. In *3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2017.

[61] T. Liu, A. Hojjati, A. Bates, and K. Nahrstedt. AliDrone: Enabling Trustworthy Proof-of-Alibi for Commercial Drone Compliance. In *IEEE 38th International Conference on Distributed Computing Systems*, 2018.

[62] Amazon Logistics. Amazon Logistics—Delivery Services Partners Program. https://logistics.amazon.com/marketing/opportunity.

[63] J. McClean, C. Stull, C. Farrar, and D. Mascareñas. A Preliminary Cyber-Physical Security Assessment of the Robot Operating System (ROS). In *Unmanned Systems Technology XV*, volume 8741. International Society for Optics and Photonics, 2013.

[64] Ernest McCollough. *Photographic Topography*. Industrial Publishing Company, San Francisco, 1893.

[65] B. Mood and K. Butler. PAL: A Pseudo Assembly Language for Optimizing Secure Function Evaluation in Mobile Devices. . *Journal of Information Security and Applications*, 40, 2018.

[66] Angela Moscaritolo. Privacy Concerns Could Impact Alexa Device Sales This Prime Day. In *PC Magazine*, June 2019. https://www.pcmag.com/news/privacy-concerns-could-impact-alexa-device-sales-this-prime-day.

[67] B. Nassi, R. Ben-Netanel, A. Shamir, and Y. Elovici. Drones' Cryptanalysis – Smashing Cryptography with a Flicker. In *IEEE Symposium on Security and Privacy*, 2019.

[68] B. Nassi, R. Ben-Netanel, A. Shamir, and Y. Elovici. Game of Drones - Detecting Spying Drones Using Time Domain Analysis. In *5th International Symposium on Cyber-Security, Cryptology and Machine Learning*, 2021.

[69] 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*, 2021.

[70] P. Nguyen, H. Truong, M. Ravindranathan, A. Nguyen, R. Han, and T. Vu. Matthan: Drone presence detection by identifying physical signatures in the drone's RF communication. In *15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017.

[71] I. Nunes, S. Jakkamsetti, N. Rattanavipanon, and G. Tsudik. On the TOCTOU problem in remote attestation. In *ACM Conference on Computer and Communications Security*, 2021.

[72] Office of the Australian Information Commissioner. Survellience and monitoring–drones. https://www.oaic.gov.au/privacy/your-privacy-rights/surveillance-and-monitoring/drones/.

[73] Standing Committee on Economic Development and Tourism. Inquiry into Drone Delivery Systems in the Australian Capital Territory, Report 6, July 2019.

[74] Parrot. ANAFI Ai—The 4G robotic UAV, November 2021. https://www.parrot.com/assets/s3fs-public/2021-11/white-paper-anafi-ai-v1.6.pdf.

[75] A. Patra, T. Schneider, A. Suresh, and H. Yalame. ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation. In *USENIX Security Symposium*, 2021.

[76] S. T. Peddinti and N. Saxena. On the limitations of query obfuscation techniques for location privacy. In *ACM International Conference on Ubiquitous Computing*, 2011.

[77] In the Supreme Court of India, Civil Original Jurisdiction, Writ Petition (Civil) No. 494 of 2012: Justice K. S. Puttaswamy and ANR vs. Union of India and ORS, 2012. https://main.sci.gov.in/supremecourt/2012/35071/35071_2012_Judgement_24-Aug-2017.pdf.

[78] P. Qiu, D. Wang, Y. Lyu, and G. Qu. Voltjockey: Breaching trustzone by software-controlled voltage manipulation over multi-core frequencies. In *ACM Conference on Computer and Communications Security*, 2019.

[79] S. Ramesh, T. Pathier, and J. Han. SoundUAV: Towards Delivery Drone Authentication via Acoustic Noise Fingerprinting. In *5th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2019.

[80] D. Rathee, A. Bhattacharya, N. Chandran, D. Gupta, R. Sharma, and A. Rastogi. SecFloat: Accurate Floating-Point meets Secure 2-Party Computation. In *IEEE Symposium on Security and Privacy*, 2022.

[81] F. J. Rodríguez-Lera, V. Matellán-Olivera, J. Balsa-Comerón, Á. M. Guerrero-Higueras, and C. Fernández-Llamas. Message Encryption in Robot Operating System: Collateral Effects of Hardening Mobile Robots. *Frontiers in ICT*, 5, 2018.

[82] White House Briefing Room. The Domestic Counter-Unmanned Aircraft Systems National Action Plan. April 25, 2022, https://www.whitehouse.gov/briefing-room/statements-releases/2022/04/25/fact-sheet-the-domestic-counter-unmanned-aircraft-systems-national-action-plan/.

[83] ROS 2–ROS 2 documentation, the latest version of the robot operating system. https://index.ros.org/doc/ros2/.

[84] A. Rozantsev, V. Lepetit, and P. Fua. Flying objects detection from a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[85] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *USENIX Security*, 2004.

[86] 911 security. Drone Detection by 911 Security—Detect and Track drones in your Airspace with the AirGuard Software Platform. https://www.911security.com/.

[87] R. W. Sinnott. Virtues of the Haversine. *Sky and Telescope*, 68(2), August 1984.

[88] John P. Snyder. *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press, 1993.

[89] Ookla SpeedTest. https://www.speedtest.net.

[90] Jonathan Stempel. Meta's Facebook to pay $90 million to settle privacy lawsuit over user tracking. *Reuters News*, February 2022. https://www.reuters.com/technology/metas-facebook-pay-90-million-settle-privacy-lawsuit-over-user-tracking-2022-02-15/.

[91] Z. Sun, B. Feng, L. Lu, and S. Jha. OAT: Attesting operation integrity of embedded devices. In *IEEE Symposium on Security and Privacy*, 2020.

[92] Quanergy Systems. Quanergy systems to showcase powerful lidar security detection system at isc west, April 2016. https://quanergy.com/wp-content/uploads/2020/02/Quanergy-Systems-to-Showcase-Powerful-LiDAR-Security-Detection-System-at-ISC-West-4-6-2016.pdf.

[93] Thales. ScaleFlyt Remote ID: Identification and tracking for safe drone operations. https://www.thalesgroup.com/en/markets/aerospace/drone-solutions/scaleflyt-remote-id-identification-tracking-safe-drone-operations.

[94] T. Vincenty. Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations. *Survey Review*, 23, 1975.

[95] X. Wang, A. J. Malozemoff, and J. Katz. EMP-toolkit: Efficient MultiParty computation toolkit, 2016. https://github.com/emp-toolkit.

[96] Y. Wang, H. Xia, Y. Yao, and Y. Huang. Flying eyes and hidden controllers: A qualitative study of people's privacy perceptions of civilian drones in the US. *Proceedings on Privacy Enhancing Technologies (PoPETS)*, 3, 2016.

[97] R. White, G. Caiazza, H. Christensen, and A. Cortesi. SROS1: Using and developing secure ROS1 systems. In *Robot Operating System (ROS)*. Springer, 2019.

[98] R. White, D. Christensen, I. Henrik, and D. Quigley. SROS: Securing ROS over the Wire, in the Graph, and through the Kernel. *arXiv:1611.07060*, 2016.

[99] R. White, H. Christensen, G. Caiazza, and A. Cortesi. Procedurally provisioned access control for robotic systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[100] Lance Whitney. Data privacy is a growing concern for more consumers. In *Tech Republic*, August 2021. https://www.techrepublic.com/article/data-privacy-is-a-growing-concern-for-more-consumers.

[101] Sarah Whittaker. New Draft Drone Laws for France Require Remote ID And Signal, April 2018. https://dronebelow.com/2018/04/12/new-draft-drone-laws-for-france-require-remote-id-and-signal/.

[102] Wikipedia. Google Street View privacy concerns. https://en.wikipedia.org/wiki/Google_Street_View_privacy_concerns (note: contains pointers to cases in numerous countries banning or strictly regulating Google Street View).

[103] Google Wing. Privacy Policy – Global. https://wing.com/privacy-global/.

[104] Google Wing. Privacy Policy - United States: Information we collect/Images, April 2021. https://wing.com/privacy-us.

[105] K. Yang, J. Yau, L. Fei-Fei, J. Deng, and O. Russakovsky. A study of face obfuscation in ImageNet. In *International Conference on Machine Learning*, 2022.

[106] A. Yao. How to generate and exchange secrets. In *IEEE Symposium on the Foundations of Computer Science*, 1986.

[107] Y. Yao, H. Xia, Y. Huang, and Y. Wang. Privacy mechanisms for drones: Perceptions of drone controllers and bystanders. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2017.

[108] S. Zahur and D. Evans. Obliv-C: A Language for Extensible Data-Oblivious Computation. In *Cryptology ePrint Archive: Report 2015/1153*, November 2015.

Code availability. Available at DOI: 10.5281/zenodo.6442206.

APPENDIX

A1 Pd-Mpc using Garbled Circuits

It is natural to ask whether Algorithm 1 can be encoded in MPC using garbled ciruits. As already mentioned in §3.2, we observed that the sizes of the circuit denoting the MPC encoding in Algorithm 2 using a GC-based setup were 10-100× larger than the circuit sizes in a secret-shared setup. For n=1000, the size of the secret-shared setup using MOTION [20]) is about 7MB. In contrast, the size of the garbled circuit using EMP [95] was 685MB. Thus, the MPC encoding shown in Algorithm 2 would be prohibitively expensive if implemented using garbled circuits.

The key problem that makes the encoding in Algorithm 2 unsuitable for GC is that the computation in lines M4-M9 is iterated *over all n drones*. Because garbled circuits require a garbling table to represent every intermediate wire, the size of the circuit is large. The circuit gets larger with larger values of n, as the loop is unrolled to create the garbled circuit.

Unlike our secret-shared implementation atop MOTION, however, traditional Yao-style garbled circuits work on Boolean representations. Thus, in a GC-based implementation atop EMP, there is no need to avoid Boolean comparisons in the MPC computation (*cf.* the discussion under "Avoiding Boolean comparisons in MPC" in §3.2). As a result, one can encode the computation so that the complex dot product calculations are performed only on drones that are in the vicinity of the citizen. We show the resulting encoding in Algorithm 3. Line GC4 ensures that the computation required for field-of-view determination is only performed on drones in the vicinity of the citizen.

We designate the citizen as the circuit garbler and the regulatory authority as the circuit evaluator. This is because the citizen can pre-compute a large number of garbled circuits and send to the regulatory authority, *e.g.*, the citizen could do so periodically, and transmit these garbled circuits when connected with sufficient network bandwidth. The live network cost between the garbler and the evaluator is only the OT cost of transmitting the garbler's inputs to the circuit. Thus, during circuit evaluation, only the citizen's inputs need to be transmitted to the regulatory authority via OT, keeping the communication requirements modest when the citizen's mobile device is on a low-bandwidth, high-latency mobile WAN. In contrast, if the regulatory authority were designated as the circuit garbler and the citizen's mobile phone as the circuit evaluator, then the OT will involve transferring city-scale data of all the drones to the mobile device on each query.

However, by only executing the computation on drones in the vicinity of the citizen, Algorithm 3 has a subtle side-channel. The side-channel arises as a result of the control-dependency between lines GC5-GC8 and the conditional on line GC4. The side-channel is akin to implicit flows in the information-flow literature [36]. The regulatory authority can infer (using timing differences or the amount of network communication) the *number* of drones in the citizen's vicinity, but not their *identities*.

Algorithm 3: DetectFieldOfView adapted for MPC using Yao-style [106] garbled circuits.

```
Input: From citizen: $\Gamma_c$, $\lambda_c$, LatVicinity, LongVicinity.
Input: From regulatory authority, identity, position and $\overrightarrow{\mathbf{D}}$ vector for all
    drones: (Id$^i$, $\Gamma_t^i$, $\lambda_t^i$, $\overrightarrow{\mathbf{D}}^i$) for all drones in city ($i$=1...$n$).
Input: Publicly-known: $\theta_1, \ldots, \theta_n$ for all drones.
Output: Revealed to citizen: $\langle$DotP$^i$, NormSquare$^i\rangle$ for drones in vicinity.
GC1  [$\Gamma$diff] = LatitudeDiff($\Gamma_t^1, \ldots, \Gamma_t^n, \Gamma_c$)
GC2  [$\lambda$diff] = LongitudeDiff($\lambda_t^1, \ldots, \lambda_t^n, \lambda_c$)
GC3  for $i \leftarrow 1$ to $n$ do
GC4      if ($\Gamma$diff[$i$] $\leq$ LatVicinity and $\lambda$diff[$i$] $\leq$ LongVicinity) then
GC5          $\overrightarrow{\mathbf{C}}^i$ = Vectorize($\Gamma_t^i, \lambda_t^i, \Gamma_c, \lambda_c$)
GC6          DotP$^i$ = $\overrightarrow{\mathbf{D}^i} \cdot \overrightarrow{\mathbf{C}^i}$
GC7          NormSquare$^i$ = $(|\overrightarrow{\mathbf{D}^i}|)^2 \times (|\overrightarrow{\mathbf{C}^i}|)^2$
GC8          Result.add($\langle$Id$^i$, DotP$^i$, NormSquare$^i\rangle$)
GC9      end
GC10 end
GC11 return Result (revealed only to citizen)
     // Steps below happen on citizen's phone (not as MPC)
GC12 for each $\langle$Id$^i$, DotP$^i$, NormSquare$^i\rangle \in$ Result do
GC13     $\phi_i$ = arccos(DotP$^i$/$\sqrt{\text{NormSquare}^i}$)
GC14     if ($\phi_i \leq \theta_i$) then shortlist the drone Id$^i$.
GC15 end
```

There are known methods in the literature to suppress this implicit information-flow side-channel ensuring that the conditional on line GC4 of Algorithm 3 executes as an *oblivious if statement*, in which dummy statements are suitably introduced so that some blocks of code are executed regardless of the value of the conditional [108]. While some MPC frameworks provide such an oblivious if statement construct (*e.g.*, Obliv-C [108]), many others do not (*e.g.*, EMP does not), and the onus is on the algorithm designer to ensure that both branches of the conditional contain identical amount of computation (with identical timing and network communication). Moreover, making the conditional oblivious has the drawback of requiring dummy code blocks to be executed regardless of the conditional, and the encoding of these code blocks adds to the circuit size. This brings us back to the same problems that plague encoding Algorithm 2 using garbled circuits, as discussed earlier in this section.

As a result, Algorithm 3 chooses not to make the conditional on line GC4 oblivious. Unfortunately, this is the cost of having a practical GC implementation. As discussed earlier, the costs are prohibitive otherwise in GC—using an oblivious if statement in Algorithm 3 with code similar in timing/communication characteristics to those in lines GC5-GC8 results in a garbled circuit that is about 685MB for n=1000 drones.

The cost of having a non-oblivious if statement is that the regulatory authority learns the number of drones in the vicinity of the querying citizen via the resulting side-channel (but not the drone identities). Thus, even if the regulatory authority were to learn the number of drones that pass the if-conditional on line 4, it would just learn the number of drones in the vicinity of *some* citizen (and not a *particular* citizen, because citizens submit queries anonymously). This is a weaker privacy guarantee than the one provided by Algorithm 2, which runs the computation on all drones. For example, it may be possible for the regulatory authority to use live drone traffic density maps to estimate the approximate locations from where the citizen issues queries, which may be unacceptable in some settings (*e.g.*, see Abidi *et al.* [3] for such an attack).

We now present an evaluation of Algorithm 3 (implemented in EMP [95]) using the same experimental setup described in §3.4. As

Number	← Number of drones in citizen's vicinity →							
in city ↓		1		2		5		10
100	LAN:	81.6±19.7ms	LAN:	94.4±9.6ms	LAN:	133.8±6.1ms	LAN:	163.0±10.1ms
	WAN:	5.9±0.4s	WAN:	9.7±1.0s	WAN:	15.8±1.4s	WAN:	24.0±3.9s
200	LAN:	107.2±4.7ms	LAN:	127.2±19.2ms	LAN:	137.6±10.3ms	LAN:	196.4±20.7ms
	WAN:	7.2±0.5s	WAN:	8.1±1.5s	WAN:	15.7±1.8s	WAN:	23.5±1.7s
500	LAN:	165.8±12.8ms	LAN:	167.2±27.0ms	LAN:	221.6±29.6ms	LAN:	235.8±25.9ms
	WAN:	9.1±0.7s	WAN:	12.5±2.1s	WAN:	19.7±3.7s	WAN:	21.4±2.1s
1000	LAN:	370.2±19.6ms	LAN:	383.4±25.8ms	LAN:	457.2±27.6ms	LAN:	554.6±51.8ms
	WAN:	11.0±1.5s	WAN:	14.8±3.3s	WAN:	19.6±1.5s	WAN:	51.9±6.2s

Average RTT between the citizen and regulatory authority on our 1Gbps LAN is 0.220ms (measured with `ping`*). In WAN, the citizen is connected via a 4G mobile data network, and the regulatory authority is an instance on Azure US West-Central. The average RTT is 282ms, and the citizen's data upload speed during this experiment varied between 74KBps to 328KBps, measured with SpeedTest [89].*

Figure 7: Overall query latency at citizen's mobile device (for Algorithm 3).

Number	← No. of drones in citizen's vicinity →			
in city ↓	1	2	5	10
100	1.349	2.032	4.076	7.483
200	1.750	2.432	4.476	7.884
500	3.207	3.888	6.190	9.598
1000	5.721	6.403	8.448	12.112

Figure 8: Per-query mobile data usage (for Algorithm 3).

before, we measured the mobile data consumption on the citizen's device to pose queries to the regulatory authority. We also measured the end-to-end latency observed by the citizen when a query is issued. Figure 7 and Figure 8 report the results of our experiments (average and standard deviation reported over 5 runs).

Observe that the mobile data consumed depends upon *two* factors: the number of drones deployed in the city and the number of drones in the citizen's vicinity. The former determines the size of the garbled circuit that is generated and must be sent from the citizen to the regulatory authority (*i.e.,* the value of *n* in Algorithm 3). The latter determines the number of drones for which lines GC5-GC8 of Algorithm 3 execute. Thus, for these experiments we crafted inputs to our MPC implementation that vary the number of drones deployed in the city, and simulate a given drone density in the citizen's vicinity.

Figure 8 shows that for a given density of drones in the citizen's vicinity, mobile data usage increases as the number of drones in the city-wide deployment (*n*) increases. This is because the size of the circuit to be garbled increases proportionally with *n*, thereby resulting in a larger amount of data to be sent from the citizen's phone to the regulatory authority. The mobile data usage also includes the OT time to send the citizens inputs, however, that is a constant value for all cases. In the worst case that we simulated—1000 drones city-wide, with a dense presence of 10 drones in the citizen's vicinity—each query from the citizen's phone consumes about 12.112MB of mobile data.

This mobile data usage is roughly double compared to the mobile data usage for 1000 drones as reported in Figure 3. Also, note that Figure 3 reports the number for the *oblivious* algorithm (Algorithm 2) that executes the computation on *all 1000 drones*. In contrast, the complex computation in the GC implementation of Algorithm 3 runs *on only 10 drones* that clear the conditional on line GC4. This experiment clearly justifies our use of secret-sharing based MPC in Pd-Mpc over a garbled circuit-based implementation.

For a given city-scale drone deployment (*e.g., n*=100 drones), observe that the per-query mobile data usage increases in proportion to the drone density in the citizen's vicinity. The reader may question why we observe this trend, given that the size of the garbled circuit is fixed by the value of *n*, and does not depend on drone density. The answer to this question lines in how Algorithm 3 is implemented atop EMP, as discussed below.

Recall that EMP lacks default support for oblivious if statements, and relies on the programmer to suitably design the algorithm to ensure that any side channels due to the lack of this support are suppressed. In particular, EMP will not compile Algorithm 3 (as shown) because the predicate used in the conditional statement on line GC4 is input-dependent. EMP compiles the circuit only if the value of the predicate on line GC4 is revealed to the citizen (thereby revealing to the citizen the identities of drones in their vicinity). Further, EMP transmits the garbled tables for the wires of the circuit corresponding to lines GC5-GC8 of Algorithm 3 on-the-fly, *i.e.,* as and when the circuit evaluator exercises that branch of the computation, and not in advance.

Moreover, EMP does not directly encode the for-loop on line GC3 in the circuit, but chooses to create a circuit with the loop unrolled (which it can, because the value of *n* is a constant that is known to the circuit garbler). Thus, a different set of garbled tables is transmitted each time the computation executes lines GC5-GC8. As a result, the per-query network communication increases with the drone density. This is indeed a side-channel that allows the regulatory authority to determine the drone density in the anonymously querying citizen's vicinity (but not their identities). In summary, the various performance optimizations that are required to make a GC-based realization of Algorithm 3 practical in terms of network communication overhead in turn result in an implementation that offers weaker privacy guarantees than the secret-shared implementation of Algorithm 2, implemented in our Pd-Mpc prototype.

A2 Precision versus Arithmetic Overflows

The DetectFieldOfView works on real-valued numbers such as GPS coordinates. While modern computer programs use floating point representations (*e.g.,* as standardized in IEEE 754) of real-valued numbers, support for floating point operations in MPC frameworks is relatively sparse in most publicly-available MPC frameworks (with SecFloat [80] being a recent development in this direction). Thus, special care must be taken when implementing algorithms that use real-valued numbers atop MPC frameworks. In this section, we discuss how we handle real-valued numbers in Pd-Mpc implemented atop the MOTION framework [20].

MOTION does not support circuit representations with floating point, and instead chooses to represent and manipulate real-valued numbers using a fixed point representation atop its support for circuit operations on integers. More precisely, the MOTION framework has a C++ front-end for programmers to write the algorithm that must be evaluated using MPC, which it compiles into a circuit that is then evaluated using the MPC protocol. Any real-valued numbers used in the computation are expressed as `double`s in the C++ program (*i.e.,* a floating point representation), which the circuit compiler encodes into a 32-bit or 64-bit integer.

The encoding function $\mathbb{E}_{\#b}(x)$ works as follows. It takes two inputs, the C++ `double` floating point number x to be encoded, and $\#b$, the number of bits in the encoded integer devoted to representing the fractional part of the floating point number. The encoding function is $\mathbb{E}_{\#b}(x) = x \times 2^{\#b}$, which is implemented by left-shifting the floating point number by $\#b$ bits. The MPC circuit expresses all operations on the real-valued number on the encoded representation, and any real-valued outputs from the algorithm are decoded back into the C++ `double` representation via the corresponding decoding function $\mathbb{D}_{\#b}$ (right-shifting the result by $\#b$ bits).

This encoded representation presents two tradeoffs based on the parameters that the MPC framework uses for encoding. The first issue concerns the bitwidth of the integer representation. Using a 64-bit integer to represent encoded real-values allows us to manipulate larger real-valued numbers and perform more operations on them without loss of accuracy or the risk of an overflow (due to the fixed-point representation). However, in comparison to a 32-bit representation, it leads to much larger circuits, and therefore a greater communication overhead in MPC frameworks. Pd-Mpc uses a 32-bit integer representation.

The second issue concerns the number of bits used for the fractional part ($\#b$). If the representation uses a larger number of bits to represent the fractional part, then any arithmetic operations on the real-valued number will offer more precision. However, because the encoding function works by left-shifting the number by $\#b$ bits, successive arithmetic operations run the risk of arithmetic overflow since fewer bits are available to represent the non-fractional part of the real-valued number. This is because $\mathbb{E}(p) \times \mathbb{E}(q)$ first results in $p \times q \times 2^{2^{\#b}}$), which is right-shifted by $\#b$ bits to preserve the invariant that $\#b$ bits are used to represent the fractional part. Nevertheless, the non-fractional part must have sufficient bits to accommodate the value of $p \times q$, otherwise the computation results in an overflow. This presents an interesting tradeoff between precision and arithmetic overflow based on $\#b$. In MOTION, the programmer specifies the value of $\#b$ in the C++ program. Figure 9 illustrates this tradeoff using a sequence of multiplications on three real-valued numbers p, q and r with various values for $\#b$. We first multiply the encoded representations of p and q and then multiply the product with r. The figure also shows the decoded value of the intermediate product $\mathbb{E}(p) \times \mathbb{E}(q)$. It compares the value of the product with the expected real-value and reports the difference (as the error term). With $\#b$=6 bits, the number of bits available for the non-fractional part is not sufficient to express the product $p \times q \times r$, and the result is an arithmetic overflow.

It is challenging to *a priori* decide the value of $\#b$ to use in the Algorithm 2 that both avoids overflow and provides sufficient precision. Note that GPS coordinates are bounded in value, and we can therefore reason about the largest and smallest GPS values that arise during the computation, and choose masking factors appropriately to avoid overflows. Thus, we can avoid overflow issues (by bounding the masking factors) on line M1 and line M2 of Algorithm 2. However, the same cannot be said of the computation of the $\overrightarrow{\mathbf{C}}$ vector on line M6 followed by the multiplications implicit in the dot product computation with $\overrightarrow{\mathbf{D}}$ in line M7. Indeed, we observed that the sequence of multiplications results in arithmetic overflows for certain values of $\#b$.

Real values p = 79.234, q = 89.312, r = 89.312

$\#b$	$T = \mathbb{D}_{\#b}(\mathbb{E}_{\#b}(p) \times \mathbb{E}_{\#b}(q))$	$\mathbb{D}_{\#b}(\mathbb{E}_{\#b}(T) \times \mathbb{E}_{\#b}(r))$
4	7078/(0.020%)	632154/(0.021%)
5	7075.22/(0.018%)	631905/(0.018%)
6	7076.61/(0.0008%)	Overflow

Figure 9: Precision versus overflow tradeoff in MOTION. Error percentages of computed result compared to the real valued result appear in parantheses.

Algorithm 4: Adaptation of Algorithm 2 to avoid arithmetic overflows in MPC computation.

```
Input: From citizen: $\Gamma_c$, $\lambda_c$, LatVicinity, LongVicinity, citizen's masking
       factors (from $\mathbb{R}^+$): $c\Gamma_1$, $c\lambda_1$, . . ., $c\Gamma_n$, $c\lambda_n$, $x\alpha_1$, $y\alpha_1$ . . ., $x\alpha_n$, $y\alpha_n$.
Input: From regulatory authority, identity, position for all drones: ($\mathsf{Id}^i$, $\Gamma_t^i$,
       $\lambda_t^i$) for all drones in city ($i$=1. . .$n$), regulatory authority's masking
       factors (from $\mathbb{R}^+$): $r\Gamma_1$, $r\lambda_1$, . . ., $r\Gamma_n$, $r\lambda_n$, $x\beta_1$, $y\beta_1$, . . ., $x\beta_n$, $y\beta_n$.
Output: Revealed to citizen:
        ⟨DotP$^i$, NormSquare$^i$, NearbyLat$^i$, NearbyLong$^i$⟩ for all drones.
[$\Gamma$diff] = LatitudeDiff($\Gamma_t^1$, . . ., $\Gamma_t^n$, $\Gamma_c$)
[$\lambda$diff] = LongitudeDiff($\lambda_t^1$, . . ., $\lambda_t^n$, $\lambda_c$)
for $i \leftarrow 1$ to $n$ do
    NearbyLat$^i$ = ($\Gamma$diff[$i$] - LatVicinity) × $c\Gamma_i$ × $r\Gamma_i$
    NearbyLong$^i$ = ($\lambda$diff[$i$] - LongVicinity) × $c\lambda_i$ × $r\lambda_i$
    ⟨$\mathbf{X}^i$, $\mathbf{Y}^i$⟩ = Vectorize($\Gamma_t^i$, $\lambda_t^i$, $\Gamma_c$, $\lambda_c$)
    $\mathbf{X}^i$ = ($\mathbf{X}^i$ + $x\alpha_i$ + $x\beta_i$); $\mathbf{Y}^i$ = ($\mathbf{Y}^i$ + $y\alpha_i$ + $y\beta_i$)
    ResultXY.add(⟨Id$^i$, X$^i$, Y$^i$⟩)
    Result$\Gamma\lambda$.add(⟨Id$^i$, NearbyLat$^i$, NearbyLong$^i$)⟩)
end
return ResultXY (value revealed to both parties)
return Result$\Gamma\lambda$ (revealed only to citizen)
------------------------------------------------------------
Analyze ResultXY and re-encode all $\mathbf{X}^i$, $\mathbf{Y}^i$ in the clear (i.e., not in the MPC
  framework)
------------------------------------------------------------
for $i \leftarrow 1$ to $n$ do
    $\mathbf{C}^i$ = ⟨($\mathbf{X}^i$ - $x\alpha_i$ - $x\beta_i$), ($\mathbf{Y}^i$ - $y\alpha_i$ - $y\beta_i$)⟩
    DotP$^i$ = $\overrightarrow{\mathbf{D}^i} \cdot \overrightarrow{\mathbf{C}^i}$
    NormSquare$^i$ = $(|\overrightarrow{\mathbf{D}^i}|)^2 \times (|\overrightarrow{\mathbf{C}^i}|)^2$
    ResultDotPNorm.add(⟨Id$^i$, DotP$^i$, NormSquare$^i$⟩)
end
return ResultDotPNorm (revealed only to citizen)
------------------------------------------------------------
// The steps below happen on the citizen's phone (not MPC)
Result = Join Result$\Gamma\lambda$ and ResultDotPNorm using Id values.
for ⟨Id$^i$, DotP$^i$, NormSquare$^i$, NearbyLat$^i$, NearbyLong$^i$⟩ ∈ Result do
    if (NearbyLat$^i$ ≤ 0) and (NearbyLong$^i$ ≤ 0) then
        $\phi_i$ = arccos(DotP$^i$/$\sqrt{\text{NormSquare}^i}$)
        if ($\phi_i \leq \theta_i$) then shortlist the drone Id$^i$.
    end
end
```

To avoid these overflows, our implementation of Algorithm 2 avoids a sequence of multiplications using the same value of $\#b$. In particular, for a product $p \times q \times r$, we observe that we can use one value of $\#b$ to compute S=$\mathbb{E}_{\#b}(p) \times \mathbb{E}_{\#b}(q)$, analyze the resulting product's value after decoding T=$\mathbb{D}_{\#b}(S)$, and then choose a different value of $\#b$ to encode T and r to compute the final product. For the example shown in Figure 9, we observe that $\mathbb{D}_9(\mathbb{E}_9(p) \times \mathbb{E}_9(q))$ = 7076.62, which differs from the real value by 0.001%, with $\#b$=9. For the second product, we encode 7076.62 and r with $\#b$=5, to obtain $\mathbb{D}_5(\mathbb{E}_5(7076.62) \times \mathbb{E}_5(r))$ = 632031, which differs from the real value by 0.001%. Note that with the same value of $\#b$=6 used for both products, there was an arithmetic overflow in the computation (Figure 9), which we avoided by using different values for $\#b$ as we progressed through the sequence of multiplication operations. We estimate a suitable value of $\#b$ for the second multiplication by analyzing the value of the first product and avoid an overflow.

We implemented the same idea in Algorithm 2. In particular, we chose to terminate the MPC computation after the computation of the $\overrightarrow{\mathrm{C}}$ vectors, and decode the x- and y-components of these vectors back into their C++ `double` representations within MOTION (thus revealing them outside MPC). We can then observe the values of the decoded `double` valued representation in C++ and decide afresh the value of $\#b$ for the subsequent dot product computation. Any further computation with the freshly re-encoded $\overrightarrow{\mathrm{C}}$ vectors will therefore avoid overflow issues that arise as a result of a sequence of multiplications on encoded values.

Unfortunately, this poses a fresh challenge. Lifting the values of the intermediate products back to C++ necessarily reveals the values to the citizen and the regulatory authority (used to determine the value of $\#b$ for the subsequent computation), which compromises the privacy of the computation. To avoid revealing the values, we therefore add masking factors just before the x- and y-components of $\overrightarrow{\mathrm{C}}$ are revealed (to both parties), and terminate the first phase of the MPC computation. This happens on line MFP7 of Algorithm 4. Masking with the $x\alpha$, $y\alpha$, $x\beta$, $y\beta$ values still involves arithmetic operations, and therefore the risk of overflow. However, these are addition operations, and therefore are far less likely to overflow as compared to multiplication operations. On line MFP13, we re-encode the (masked) values for the rest of the calculation. This step encodes the masked `double` values into the fixed-point 32-bit representation using the new value of $\#b$. We then resume the MPC protocol with re-encoded values, which unmasks and recovers the raw value of the $\overrightarrow{\mathrm{C}}$ vectors on line MFP15. Following this step, Algorithm 4 then proceeds identically to Algorithm 2.

A3 Security Analysis of Pd-Ros

This section analyzes the security of the Pd-Ros design described in §4.2. Recall that our design is tailored to allow well-intentioned ROS applications to establish that they comply with privacy by: ⓐ ensuring that sensitive data that they publish are consumed only by the trusted sanitizer application running in the secure world; and ⓑ ensuring that they only subscribe to data that has been sanitized by the trusted video sanitizer.

Pd-Ros relies on two key security mechanisms:

① *Attestation reports.* Pd-Ros relies on boot-time attestation as well as runtime attestation of applications as they are launched. Boot-time attestation ensures the integrity of the secure world when the drone is initially booted up. The secure world also performs an integrity measurement of the normal world when it boots the normal world. Further, the design of Pd-Ros measures the integrity of the normal world, ROS, SROS, Sanitizer-FE as well as the application launcher and stores the corresponding measurement in the secure world via the trusted audit logger. A security analyst can use these measurements in the audit trail to determine that the normal world, ROS, SROS, Sanitizer-FE and the application launcher are untampered at least as of the time of measurement, *i.e.,* when an application is launched in the normal world. We rely, as is standard in all attestation protocols [85], on the integrity measurements being rooted (transitively) in trusted hardware, in our case, the ARM TrustZone.

② *Verifying flow redirection.* The application launcher stores a copy of the manifest of the application that it launches in the secure world. SROS enforces that all ROS applications communicate in accordance with their declared manifests. Manifests are part of each application's X.509 certificates, and the SROS application launcher verifies the certificate of the application during launch, thereby ensuring that neither the application nor the manifest have been tampered with.

Once the manifests are stored in the audit trail, a security analyst can use the stored manifests to determine that: ⓐ if an application publishes sensitive data, which we identify by their topic names, *e.g., VideoFeed*, then no application other than the trusted sanitizer's front-end subscribes to that topic; and ⓑ if an application must consume sensitive data, then it only consumes the sanitized data stream published by the trusted video sanitizer, *e.g., PrivVideoFeed*.

Given the above discussion, the security provided by Pd-Ros relies on the following factors:

- **(F1)** *Reliability of attestation reports.* Pd-Ros verifies the integrity of the normal world, ROS, SROS, Sanitizer-FE and the application launcher via the attestation reports. However, trusted hardware-based attestation protocols are known to be vulnerable to time of check to time of use (TOCTTOU) problems, and our Pd-Ros prototype is no different. An attacker can use zero-day exploits against either the normal world, ROS, SROS, Sanitizer-FE or the application launcher causing their runtime behavior to differ from that of the attested version. As a result, it may be possible to bypass SROS enforcement of application manifests using such TOCTTOU attacks. That said, Pd-Ros can incorporate recently-proposed methods to address the TOCTTOU problem in remote attestation [71]. Pd-Ros could also use fine-grained path-based attestation of application functionality (*e.g.,* C-FLAT [2] or OAT [91]), which attests the precise execution path followed by the application, thereby providing protection against runtime exploits on the application.
- **(F2)** *Faithfulness of manifests in capturing inter-application communication.* Pd-Ros relies in a key way on the runtime enforcement of application communication patterns based on the publish/subscribe topics declared in the SROS application manifests. This assumption is satisfied so long as ROS applications only communicate via ROS abstractions, *i.e.,* topics. However, it is known [16] that applications can bypass ROS abstractions, and communicate directly via low-level abstractions, *e.g.,* sockets or shared memory, by invoking raw system calls. The ROS community has addressed this problem by building MAC enforcement of the application communication patterns implied by the manifests within the operating system at the level of processes implementing the ROS applications

(*e.g.,* the Privaros [16] or the SROS+AppArmor[9] projects). Pd-Ros could also use these methods to enhance the normal world operating system, and obtain protection from attacks that bypass ROS communication abstractions.

An alternative approach is for an auditing authority to verify the ROS applications running on the drone via static analysis. The drone operator can provide the application binaries to the citizen (or any other auditing authority), who then verifies, using the attestation reports, that those were the binaries that were launched on the drone. The auditing authority can then use static analysis on each application binary to ensure the absence of low-level calls (*e.g.,* system calls to open sockets or establish shared memory) in the application binary. Applications that pass this analysis can therefore only use SROS for communication, and the corresponding SROS manifests therefore suffice to capture inter-application communication patterns. This approach has the advantage of imposing no additional runtime performance overheads, unlike the MAC enforcement systems discussed in the previous paragraph.

- (**F3**) *Ability of trusted sanitizers to enforce privacy.* Finally, Pd-Ros relies on domain-specific sanitizers to enforce privacy. For example, it relies on the video sanitizer to locate all privacy-sensitive objects in each frame (*e.g.,* human faces or vehicle registration plates), and suitably distort them so that the sanitized feed respects privacy. Since the trusted sanitizer is part of the trusted-computing base, we do not attempt to further verify the functioning of the sanitizer in our Pd-Ros design. A pair of malicious ROS application could also attempt to leak privacy-sensitive objects via a low-rate side-channel that still passes through the trusted sanitizer. For example, a malicious source application could signal the presence of a particular individual in the video feed using a single bit in the header of the video stream. If this bit is not detected and sanitized by the trusted sanitizer, a colluding malicious downstream target application can infer the presence of the individual as indicated by the malicious source application. Such side-channel attacks are currently out of scope for Pd-Ros, and new methods need to be developed to detect such side-channels.

Finally, we note that the security analysis above applies to the design of Pd-Ros discussed in §4.2, which was presented assuming that the underlying ARM TrustZone SoC lacks support for the TrustZone Peripheral Controller (TZPC) [11]. If the SoC indeed supports TZPC, then Pd-Ros can use the following architecture with sensitive data confined to the secure world, and only leaving the secure world in sanitized form. With the TZPC, TrustZone offers the ability to securely split peripherals between the secure and the normal worlds. A peripheral can be assigned to the secure world for exclusive access, and therefore cannot be accessed by applications running within the normal world. Prior work [59] has used this to enable a trusted input path via secure world control of certain peripherals (*e.g.,* touchscreen).

On a TZPC-equipped SoC it is possible to accomplish goal **G1** as follows. The camera can be exclusively assigned to the secure world, and a trusted application executing in the secure world can sanitize the video feed before it is consumed by applications in the normal world. Any applications that require access to the raw video feed (*e.g.,* a navigator that requires sharp video frames) would also execute within the secure world after *a priori* vetting that they do not leak the raw footage. This would ensure that the camera data never leaves the secure world unsanitized, and the drone simply has to prove the existence of the above setup to a querying citizen, which can be accomplished by implementing secure boot and runtime attestation of the secure world.

The precise notion of what it means to "sanitize" a video footage to preserve privacy is region-specific, and beyond the scope of Privadome. In this paper, we intentionally do not commit to any particular method as an acceptable notion of video sanitization. For example, it could mean that the footage is obtained at low resolution. Or it could mean that sensitive objects, such as faces and car registration plates identified in the video feed, are identified and blurred (*i.e.,* pixelated). This notion has been used in prior work [4, 16], and is also the approach that is employed to preserve citizen privacy in Google Street View. In fact, prior work has even developed (MPC-based) methods to allow individual citizens to specify their own privacy policies, *e.g.,* to have just their appearance blurred or edited out of the footage altogether [4]. All of these are viable options within the broader Privadome-framework, but for the sake of having a concrete policy for our discussion, we consider blurring frames (or *all faces* within a frame) as our video sanitization policy. With TZPC, a trusted application that identifies and blurs faces in video feeds would achieve Pd-Ros' goals.

We do not expect all drone platforms to have the TZPC on their SoC. For example, the experimental platform (a NVidia Xavier NX development board) that we use to build Pd-Ros does not offer exclusive secure world access to peripherals. We therefore focus on how to achieve Pd-Ros' goals even if the SoC lacks TZPC support. In the absence of TZPC, the video footage from the camera is accessible in the normal world. The mechanisms introduced by Pd-Ros must therefore be tailored to the software environment executing in the normal world.

With this TZPC-based design, the factors **F1** and **F2** discussed above do not apply (disregarding zero-day exploits against the secure world itself, as is standard, because it is part of the trusted-computing base). In this design, the only security consideration would be factor **F3**, *i.e.,* that of ensuring that the trusted sanitizer indeed removes all occurrences of sensitive objects from the video feed. It must not come as a surprise that the Pd-Ros design that leverages TZPC hardware support offers stronger security guarantees (or relies on the security of fewer components) than when such support is not available.