AliDrone: Enabling Trustworthy Proof-of-Alibi for Commercial Drone Compliance

Tianyuan Liu, Avesta Hojjati, Adam Bates and Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign, Champaign, US
{tliu60, hojjati2, batesa, klara}@illinois.edu

***Abstract*—**
Commercial use of Unmanned Aerial Vehicles (UAVs), or *drones*, promises to revolutionize the way in which consumers interact with retail services. However, the further adoption of UAVs has been significantly impeded by an overwhelming public outcry over the privacy implications of drone technology. While lawmakers have attempted to establish standards for drone use (e.g., *No-Fly-Zones (NFZs)*), at present a general technical mechanism for policy enforcement eludes state-of-the-art drones.

In this work, we propose that Proof-of-Alibi (PoA) protocols should serve as the basis for enforcing drone privacy compliance. We design and implement AliDrone, a trustworthy PoA protocol that enables individual drones to prove their compliance with NFZs to a third party Auditor. AliDrone leverages trusted hardware to produce cryptographically-signed GPS readings within a secure enclave, preventing malicious drone operators from being able to forge geo-location information. AliDrone features an adaptive sampling algorithm that reacts to NFZ proximity in order to minimize the processing cost. Through laboratory benchmarks and field studies, we demonstrate that AliDrone provides strong assurance of geo-location while imposing an average of 1.5% overhead on CPU utilization and 0.3% of memory consumption. AliDrone thus enables the further proliferation of drone technology through the introduction of a trustworthy and accountable compliance mechanism.

***Index Terms*—Drone, GPS Forgery Attack, Privacy, TrustZone**

I. Introduction

The Unmanned Aerial Vehicles (UAVs) technology, also known as "drones", enables many promising applications. Besides military purpose, many businesses are paying more attention on the commercial usage of drones. For example, Amazon announced its Air Prime Delivery Service [1] in 2013, aiming to deploy small drones to deliver lightweight packages. The expected delivery time can be as short as 30 minutes after the purchase is made, which is much faster than the best delivery option in the current state. Additionally, other drone applications include infrastructure construction, precision agriculture, and photography [2]–[4].

Despite all the benefits of drones, the public has shown great concern of privacy for the drone applications. A drone equipped with high resolution camera can surreptitiously surveil anyone's backyard 400 feet high in the air. Since 2010, the Federal Aviation Administration (FAA), has been working on the UAV regulations to control the risks of commercial drone usage. The most recent rules [5] include requirements on the pilots, the UAV specs, and the locations where drones are allowed to fly. However, these rules mainly focus on the safety protection but fail to defend against the privacy violation.

One promising countermeasure for mitigating drone surveillance is the establishment of no-fly-zones (NFZs) over privacy-sensitive locations. If a drone is sufficiently far away from a sensitive area, surveillance cannot be carried out successfully. The FAA has designated a variety of NFZs, primarily for safety purposes, around critical infrastructures such as airports. An established NFZ specifies that no drone is permitted to fly within 5 miles of the protected location. To more effectively notify drone operators of NFZs in their area, the FAA has even published the B4UFLY mobile app [6]. Unfortunately, regulation alone cannot prevent drones from flying over restricted areas; as the drone navigates in open airspace, it is hard for an observer on the ground to accurately determine the location of a drone. Instead, what is needed is a reliable means of tracking drone locations for the detection of NFZ policy violations.

In this paper, we present the design and implementation of AliDrone, a geo-location based alibi protocol that enables drones to generate proof-of-non-entrance to an NFZ. We define three roles in the system: *Zone Owners* that own some property, *Drone Operators* that operate a drone and control its navigation through an area, and *Auditors*, authorized third parties (e.g., local agents of the FAA) that attest drone locations and detect any non-compliance on NFZ regulations. Before flying, the Drone Operator queries the Auditor for the location of nearby NFZs. While flying, the drone computes an *alibi*, i.e., a signed GPS trace, based on its real time location. At the end of the flight, the Drone Operator submits the drone's Proof-of-Alibi (PoA) to the Auditor. The Auditor then verifies the PoA and initiates punishment on the Drone Operator if a policy violation is detected.

We design AliDrone with consideration that a *Dishonest Drone Operator* may try to navigate the drone over a restricted area without being detected by the Auditor. Such an attacker could attempt to forge an innocent compliant route and compute its alibi based on this forged GPS trace. As a result, an adversary may take a shortcut route or gain pictures of the restricted area.

Defending against such adversary is challenging. As the owner of the drone, the Dishonest Drone Operator has privileged access to the drone software stack as well as any

Funded by the Department of Energy, Award Number DE-OE0000780.

exposed hardware, meaning the attacker could attempt to extract security keys used in the alibi protocol or replace the system components with malicious software.

Our design relies on the existing secure hardware to provide a trusted execution environment for drones to generate their Proofs-of-Alibi (PoA). The PoA is a keyed cryptographic hash of the drone's GPS trace that is signed by a security key protected by secure hardware. Thus, the Drone Operator does not have access to the private signing key. We require the Drone Operator to submit the alibi (i.e., the GPS trace) along with this proof to the Auditor. The Auditor knows each drone's public key and can therefore verify the signatures. In this way, the Drone Operator is unable to tamper with the alibi submitted to the Auditor.

We outline the paper's main contributions below:

- We present AliDrone, a lightweight and practical alibi system that enables drones to generate trustworthy proof of privacy compliance.
- We introduce an adaptive sampling mechanism to minimize the processing and energy overhead.
- We provide performance benchmarks and perform an exhaustive real-world evaluation of AliDrone.

The rest of this paper is organized as follows:

- Section II, describes the background of drones and secure hardware technology;
- Sections III and IV, demonstrates the system model followed by the design decisions;
- Section V, describes the hardware platform and implementation details;
- Section VI, presents the evaluation of AliDrone;
- Section VII, discusses limitation and extensions of AliDrone;
- Section VIII, is dedicated to related work.

II. Background

A. Unmanned Aerial Vehicle (UAV)

An unmanned aerial vehicle, usually referred to as a drone, is an aircraft without a human pilot onboard. Such devices can be controlled remotely by the operator within a distance of 200 - 3,000 meters. A typical drone costs from $200 to $1,000. It flies at up to 40mph with a flight time of 20 - 30 minutes. Most drones are equipped with a camera, which enables many popular applications such as aerial photography. Recently, some drones with programmable features are available on the market. These drones can be programmed to perform actions including object tracking, navigation, and surveillance [7].

B. Trusted Execution Environments

Trusted Execution Environment (TEE) is a set of security extensions added to main processors. These processors partition the hardware and software and run a separated subsystem known as "secure world" in addition to the normal operating system, a.k.a. "normal world". The TEE technology is programmed into the hardware to protect the memory and peripherals. Consequently, security is enforced without degrading the system performance. TEE can be implemented on commercial secure hardware such as ARM TrustZone [8] and Intel SGX [9].

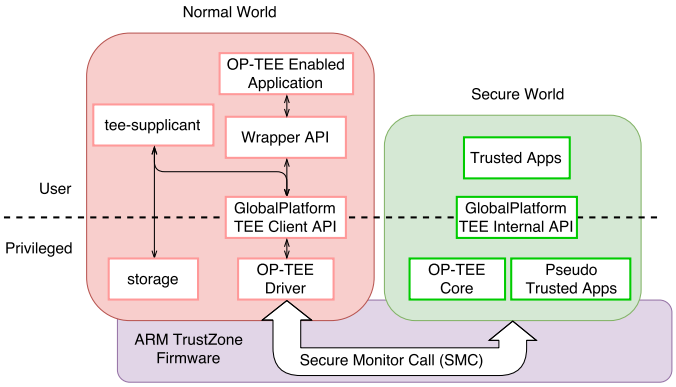

Fig. 1. OP-TEE Architecture. The code and data in secure world are protected by hardware. The switching between two worlds are triggered via Secure Monitor Call (SMC).

After the initial effort in standardizing software development for TrustZone, ARM partnered with GlobalPlatform to define a new TEE API. TEE encompasses three major features:

- Safe and secure boot ensures all system software components are in a known and trusted state before launching the operating system.
- Isolated execution of critical applications in a virtualized environment.
- Data protection of trusted applications in terms of integrity and confidentiality.

In this work, we leverage the integrity feature of TEE to authenticate the geo-location data.

C. OP-TEE

OP-TEE is an open source project for TEE in Linux using the ARM TrustZone technology. It implements a TEE client in the normal world and a TEE core in the secure world using the GlobalPlatform TEE System standard. Fig. 1 shows the architecture of OP-TEE.

OP-TEE provides a minimal secure kernel (OP-TEE core) which can be run in parallel with a normal world OS such as Linux. It provides drivers (OP-TEE Driver) for the normal world OS to communicate with the secure world. The transition between the two worlds are triggered via Secure Monitor Calls (SMC). It uses a daemon service in the normal world, i.e., tee-supplicant, to help the Trusted OS with the miscellaneous such as storage access.

OP-TEE allows two types of Trusted Applications (TAs) [10]. A normal TA runs in non-privileged mode in the secure world. When compiled, a TA is signed by a private key which is unknown to the user in the normal world. Hence, it can be stored in the untrusted storage. Every TA is assigned a unique UUID. When an OP-TEE enabled application calls an interface provided by a specific TA, it provides the associated UUID and the interface ID. Then, the tee-supplicant will locate the TA by the UUID in the storage and help the OP-TEE

core to load the TA. Dynamically loading the normal TAs can reduce the size of TEE core. However, such TAs cannot access the devices and peripherals by their physical addresses. The other type of TA is called Pseudo Trusted Application (PTA). Unlike the normal TAs which are dynamically loaded when necessary, PTA are statically built into the OP-TEE core. PTA can access the peripherals by creating a mapping from the physical address to the memory. In this work, our design involves in both TA and PTA components.

III. System Model

A. Physical Model

We consider a *Drone Operator* that instructs a drone to navigate a given flight pattern. We represent the drone's activity as a series of samples $S = (lat, lon, t)$, each represented as a tuple of latitude, longitude and timestamp that are sampled from a GPS receiver. A particular drone flight pattern F can thus be summarized as:

$$F = \{S_0, S_1, \ldots, S_n\}.$$

This work considers a situation where a drone must navigate an area in which many NFZs are present. We assume all NFZs to be circular, and are defined by:

$$z = (lat, lon, r),$$

where lat and lon are the latitude and longitude of the center, and r is the radius of the circle. We refer to the entities who own the NFZs as *Zone Owners* throughout the rest of this paper. If a drone passes into an NFZ, we say that the privacy of that Zone Owner is violated.

We assume that each drone is associated with an identifier, similar to a vehicle license plate, which is visible by an observer on the ground. If a Zone Owner spots a drone close to her NFZ, she may suspect that privacy violation has occurred. The Zone Owner will record the drone ID and report the incident to an *Auditor*, which is an authorized third party, e.g., a local Federal Aviation Administration (FAA) agent. The Auditor uses the drone ID to recover the flight pattern F from the Drone Operator, then determines if the privacy violation did occur. *In our model, the burden of proof rests on the Drone Operators to prove conclusively that their drones could not have been present in the NFZs.*

If F is insufficient to produce Proof-of-Alibi, the Auditor concludes that a privacy violation has occurred. The Auditor will then initiate punitive measures against the Drone Operator. The punishment for privacy violation is orthogonal to the purpose of this work, and can be specified through policy or legislation.

B. Threat Model

We consider the adversary as a dishonest Drone Operator (or rogue drone) that wants to violate NFZ airspace without being detected by the Auditor. Such an adversary may be small business looking to reduce costs by taking a shortcut, or a journalist or amateur operator attempting to acquire footage from a restricted area [11]. To avoid detection, the adversary will attempt to forge an innocuous route to present to the Auditor in place of its actual illicit GPS trace. This feat may be attempted through pre-computing a route that does not intersect any NFZ, replaying a previously reported route, or relaying a route from another drone. We use the term *GPS forgery attack* to denote this attack in the rest of the paper.

We assume the presence of secure hardware within the drone that provides a trusted execution environment (i.e., ARM Trustzone, Intel SGX). Furthermore, we assume that an asymmetric sign key pair is generated within TEE by the hardware manufacturer, and the private key is not known by the Drone Operator. Side channel attacks on the enclaves [12]–[14] are not considered in this work. While the attacker can attempt to install malicious software on the drone platform, we assume the correctness of the GPS hardware. We also do not consider GPS spoofing attacks in which the GPS receiver is manipulated from the ground through the broadcast of incorrect GPS signals [15], [16]; such attacks can be mitigated through existing defenses [17]–[20].

IV. System Design

A. Design Goals

Our system protects the privacy of Zone Owners by allowing them to request no-fly-zones (NFZs) upon their properties. The solution enables the drones to present trustworthy, geo-location based proof-of-alibis (PoAs) proving that the drones do not fly over the NFZs. The PoA will be verified by a trusted third party, known as the *Auditor*. Before we describe the design decisions in detail, we list our design goals as follows:

- **G1 Completeness:** The PoAs generated by the drone must prove that it does not fly over *any* NFZ during the *entire* flight period.
- **G2 Low Overhead:** The computation of trustworthy PoAs should impose *small* processing overhead for the drones.
- **G3 Unforgeability:** The Auditor *must not accept* any PoA if it is forged by Drone Operator.

B. Protocol Overview

As described in section III, our system involves three entities: a Drone Operator, a Zone Owner and an Auditor. Fig. 2 demonstrates the interactions among these entities. We describe the high level protocol in this section. A summary of cryptographic keys and data used by the protocol is presented in Table I.

0. Drone Registration: We require that a drone should be registered at the Auditor before operated in the field. The Drone Operator will generate an asymmetric keypair $\mathcal{D} = (D^+, D^-)$ and provide the public key D^+ to the Auditor. To enable trustworthy report of geo-locations, we require that an asymmetric keypair for the Trusted Execution Envirionment (TEE) on the drone $\mathcal{T} = (T^+, T^-)$ is generated at manufacturing time. The TEE sign key T^- is only accessible by TEE and the verification key T^+ is known to the drone owner when the device is merchandised. At registration, the TEE verification key T^+ should also be submitted to the Auditor. An identifier

Notation	Description	Knowledge
id_{drone}	Identifier of drone. It must be carried on the drone during operation.	All parties
id_{zone}	Identifier of NFZ. Associated with the latitude, longitude and radius of the property.	All parties
T^-	Private TEE sign key. Used to sign GPS data in TEE.	Drone TEE
T^+	Public TEE verification key. Enables verification of signed GPS data.	Drone Operator/Auditor
D^-	Private sign key of a Drone Operator. Used to authenticate zone query messages. Associated with id_{drone}.	Drone Operator
D^+	Public verification key of a Drone Operator. Enables verification of signed zone query messages.	Auditor

Table I
Notations of keys and data used by AliDrone protocol. Column **Knowledge** indicates the parties who have access to the information.

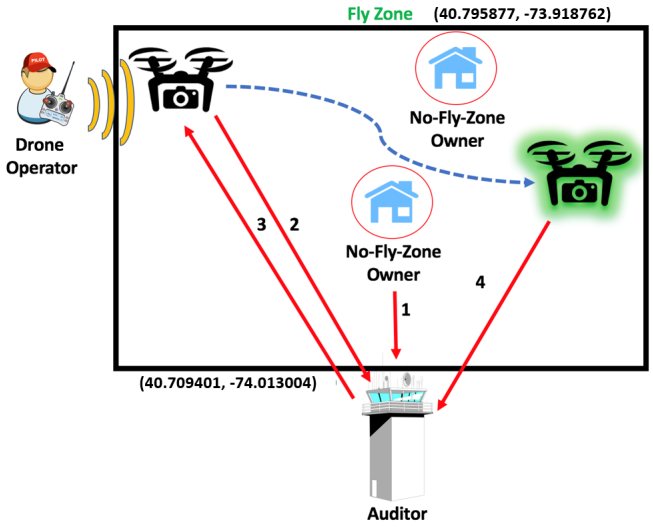

Fig. 2. An overview of system workflow. The process starts where the Zone Owner submits the coordinates to the Auditor (task 1). Then, Drone Operator submits its flight plan to the Auditor and in response receives the NFZs within the flight zone (task 2 and 3). After the flight, Drone Operator provides the proof-of-alibi, showing that it has not flown over the NFZs to the Auditor (task 4).

id_{drone} will be then issued to the drone. This identifier is similar to a vehicle license plate, which must be carried on the drone when it operates. Therefore, an entry of registered drone can be expressed as $(id_{\text{drone}},\ D^+,\ T^+)$.

1. Zone Registration: In order to register an NFZ, a Zone Owner submits to the Auditor the coordinates and radius of the property, i.e., $z = (lat, lon, r)$, as well as a proof of ownership. Upon request approval, the Auditor will issue an identifier id_{zone} to the Zone Owner and add a new entry (id_{zone}, z) to the NFZ database.

2-3. Zone Query/Response: Before a drone starts navigation, the Drone Operator should query the auditor for the NFZ information. The query is comprised of the drone id, two GPS coordinates (x_1, y_1) and (x_2, y_2), indicating a rectangular navigation area, and a random nonce signed by the drone sign key D^-, i.e.,

$$(id_{\text{drone}},\ (x_1, y_1),\ (x_2, y_2),\ \text{nonce},\ \text{Sig}(\text{nonce}, D^-)).$$

The Auditor first checks if the query is sent from a registered drone by verifying the signature on the nonce. Then, it pulls a list of NFZs $\{z_1, z_2, \cdots, z_m\}$ within the rectangle and responses with the coordinates and radii of the zones. The drone can use the NFZ information to compute a viable route to its destination.

4. Proof-of-Alibi Submission: During the flight, the drone computes the Proof-of-Alibis (PoAs) and persists the PoAs to storage. The purpose of the PoA is to show that the drone does not enter any NFZ during the flight. The detailed design of PoA is presented in section IV-C. At the end of the flight, the Drone Operator must submit the PoAs to the Auditor for verification. To enable real-time auditing, the drone could alternately transmit its PoAs in real-time to the Auditor; however, we do not pursue this solution in our work as it would increase battery drain, violating Goal G2.

C. Trustworthy Proof-of-Alibi

In this section, we introduce the concept and design of Proof-of-Alibi (PoA), which enables drones to generate unforgeable GPS traces. We first explain how the geo-location information serves as a proof of privacy compliance (Goal G1). Then, we demonstrate an extension in the trusted execution environment (Goal G3) and an optimization to reduce processing overhead (Goal G2).

1) Possible Traveling Range: To prove that a drone does not enter an NFZ, we show that it is physically impossible to travel into the zones based on its geo-locations. The idea of this proof is based on the fact that drones have a maximum traveling speed $v_{\max}$, which is restricted to 100 mph by the FAA regulation [5]. This enables the computation of the *possible traveling range* using two GPS coordinates.

Consider that the drone produces two GPS samples $S_1 = (x_1, y_1, t_1)$ and $S_2 = (x_2, y_2, t_2)$. Denote the location of the drone at arbitrary time $t \in [t_1, t_2]$ as (x, y), the possible traveling range can be described as an ellipse $\mathcal{E}$ with (x_1, y_1) and (x_2, y_2) being the two focuses:

$$\mathcal{E}(S_1, S_2) = \{(x, y) \mid d_1 + d_2 \leq v_{\max}(t_2 - t_1)\},$$

where $d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$.

Suppose the drone operates near an NFZ $z = (x_0, y_0, r_0)$. The GPS samples (S_1, S_2) can prove that the drone does not enter zone z during (t_1, t_2) if the ellipse does not intersect with the circle representing zone z. Otherwise, it suggests that the drone may travel into zone z during $[t_1, t_2]$.

During the flight, we require the drone to collect a set of GPS samples and define the set of the samples as *alibi*:

$$\text{alibi} := \{S_0, S_1, \cdots, S_n\}.$$

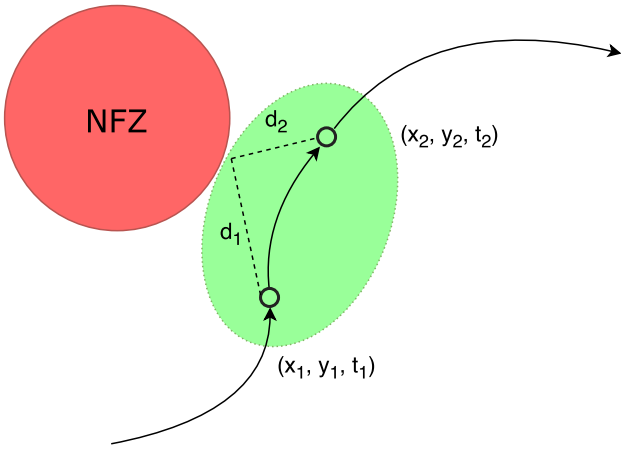

Fig. 3. Possible traveling range and a single NFZ. The possible traveling range should not intersect with the NFZ to produce sufficient alibi.

Given a set of NFZs $Z = \{z_1, z_2, \cdots, z_m\}$, we say that the alibi is *sufficient* if every pair of two consecutive GPS samples prove impossibility of traveling into all the NFZs, i.e.,

$$\mathcal{E}(S_i, S_{i+1}) \cap \left(\bigcup_{z \in Z} z \right) = \varnothing, \ \forall\, i < n. \tag{1}$$

Otherwise, we say the alibi is *insufficient*. Insufficient alibi suggests that the drone may travel into NFZs during the flight. Hence, it does not show compliance with the no-fly rule. Consider a simple case where only one NFZ is on the map shown in Fig. 3. The minimum sampling rate that produces sufficient alibi should results in an ellipse that is tangent to the NFZ.

2) TEE Enabled GPS Sampling: To ensure that such alibi cannot be forged by Drone Operators, our solution leverages trusted hardware to authenticate the GPS data in a Trusted Execution Environment (TEE). We move the sampling logic to the secure world to guarantee that the GPS data is collected from the GPS hardware. The GPS data is signed by the TEE sign key T^- before it leaves the secure world. We define the *Proof-of-Alibi (PoA)* as a series of GPS samples along with the TEE signatures, i.e.,

$$\text{PoA} \coloneqq \{(S_0, \mathsf{Sig}(S_0, T^-)), (S_1, \mathsf{Sig}(S_1, T^-)), \cdots\}.$$

The sign key T^- is only available to TEE such that a Drone Operator in the untrusted environment cannot forge the signatures. The verification key T^+ is known to the Auditor at registration stage, and thus the Auditor is able to detect if the GPS data is modified. Our design can be generalized to trusted hardware platforms including Intel SGX and ARM TrustZone. We present the an ARM TrustZone based architecture of AliDrone in Fig. 4.

The Auditor runs an AliDrone Server. It stores the information of registered drones and NFZs, and provides an interface to query the NFZ information to the drone client. Upon receiving the PoAs from drones, it verifies the sufficiency of the PoAs (see equation (1)). After the PoA verification, the AliDrone Server should save the PoAs for a couple of days. This is because a Zone Owner may report a violation afterwards and the PoAs will serve as evidence for the accusation.

The drone client consists of three components: GPS Driver, GPS Sampler and Adapter. GPS Driver runs in the kernel space of the secure world. It is used to access the GPS receiver and parse the raw GPS data into coordinates and timestamps.

GPS Sampler runs in non-privileged mode in the secure world. It exposes an interface `GetGPSAuth` to the Adapter to produce an authenticated GPS sample. It reads the parsed GPS data from the underlying GPS Driver and signs the data with the TEE sign key T^-.

The Adapter is a daemon service in the normal world. It has access to the GPS receiver and controls the PoA sampling rate using the adaptive sampling mechanism, which will be introduced in section IV-C3. In addition, it is responsible for encrypting the PoA with the public encryption key of the AliDrone Server.

3) Adaptive Sampling: A commercial GPS receiver can update the GPS measurements with a maximum rate of 5Hz [21]. However, performing frequent sampling in AliDrone is expensive because signature and world-switching operations are costly. Maintaining the maximum sampling rate has a non-negligible amount of processing overhead on the resource limited hardware. Therefore, an adaptive sampling mechanism is essential to minimize the processing overhead for the drones.

As mentioned in section IV-C1, two samples (S_1, S_2) are sufficient to prove alibi from zone z if the ellipse of possible traveling range does not intersect the zone, i.e.,

$$\mathcal{E}(S_1, S_2) \cap z = \varnothing.$$

Given a traveling trace described by a series of samples $\{S_0, S_1, \cdots, S_n\}$ such that $t_i < t_{i+1}$, we can conclude that

$$\mathcal{E}(S_i, S_j) \subset \mathcal{E}(S_i, S_k), \ \forall\, i < j < k.$$

This implies that if the sample pair (S_i, S_k) is sufficient, all the intermediate samples in between are not needed in the PoA. Denote the PoA as a set of samples selected from the trace $\{S_{k_0}, S_{k_1}, \cdots, S_{k_m}\}$ and let the first sample from PoA be $S_{k_0} = S_0$. The task of the Adapter is to find

$$k_{i+1} = \underset{j}{\operatorname{argmax}} \left(\mathcal{E}(S_{k_i}, S_j) \cap z = \varnothing \right), \ \forall\, k_i < j < n.$$

Since the Sampler only samples the current GPS information by demand, it can be too late to recover a previous sample when the current location already violates PoA sufficiency. Therefore, the Adapter must take a sample when the boundaries of the possible traveling range and the NFZ are close.

Consider the worst case where the drone flies towards the NFZ $z = (x_0, y_0, r_0)$ at maximum speed $v_{\max}$. Assume that the GPS receiver has a maximum update rate of R Hz. Let the last sample recorded in PoA be $S_1 = (x_1, y_1, t_1)$ and the latest sample measured by the Adapter be $S_2 = (x_2, y_2, t_2)$ such that

$$D_1 + D_2 \geq v_{\max}(t_2 - t_1) \tag{2}$$

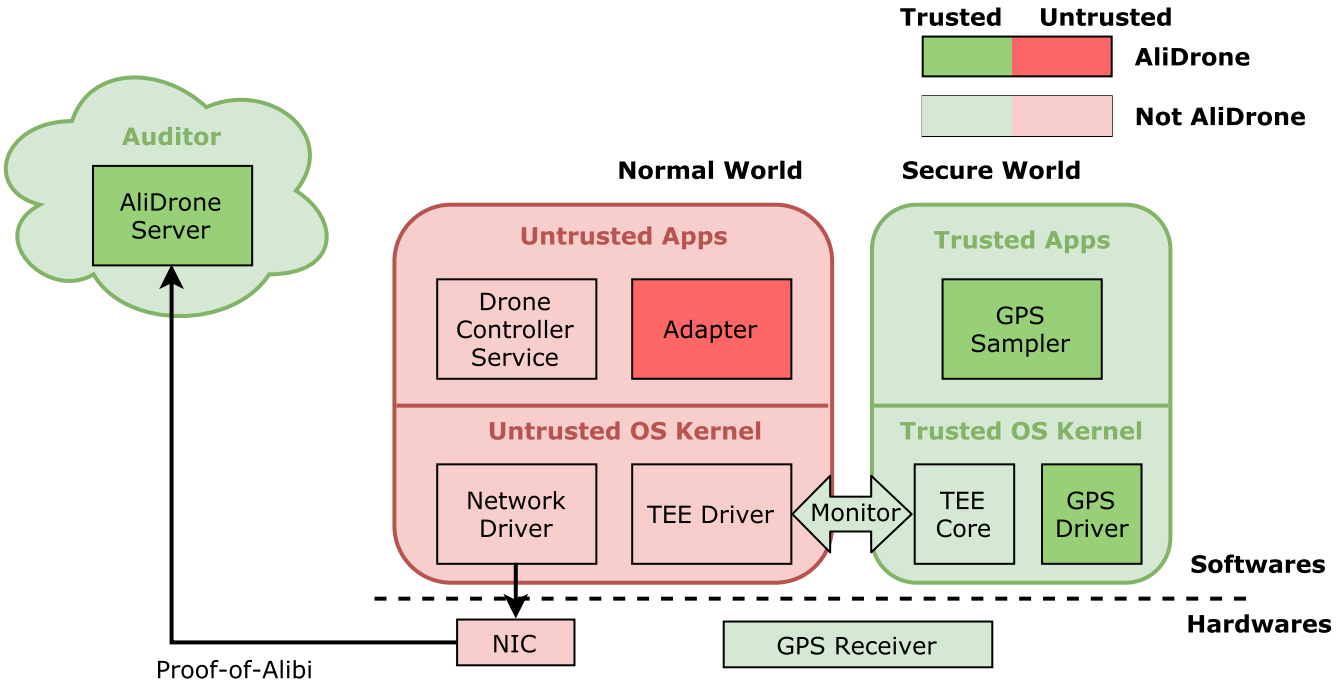

Fig. 4. AliDrone System Architecture. AliDrone enables trustworthy PoA generation on the drone by performing GPS sampling in a TEE. The GPS data is sampled, encrypted and signed by the trusted application GPS Sampler. The Adapter runs adaptive sampling algorithm and adjusts GPS sampling rate in real time. The Auditor runs AliDrone Server to verify the PoA uploaded by the drone.

where $D_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - r$ is the distance between the drone and the boundary of z. The next GPS update will be made in $\Delta t = \frac{1}{R}$ and the difference of such distance will be $\Delta D = -\frac{v_{\max}}{R}$.

The sample S_2 should be made if the next measurement will be *insufficient*, i.e.,

$$D_1 + D_2 + \Delta D < v_{\max}(t_2 - t_1 + \Delta t).$$

Therefore we have

$$D_1 + D_2 < v_{\max}(t_2 - t_1 + 2/R) \tag{3}$$

Therefore, we can conclude that a sample should be recorded in PoA if conditions (2) and (3) are both true.

When multiple NFZs are present, we only need to prove PoA sufficiency for the closest zone. We present the Adaptive Sampling algorithm in Algorithm 1. In each iteration, the Adapter first samples the GPS data in the normal world by calling `ReadGPS()` with the same rate R that the GPS receiver updates the measurements. Then, it finds the closest zone from NFZ list. If both conditions (2) and (3) hold, it calls `GetGPSAuth()`, which acquires the sample and the signature from the GPS Sampler in the secure world.

Algorithm 1: Adaptive Sampling Algorithm. The adaptation is achieved by skipping unnecessary calls of `GetGPSAuth()` interface.

NextSample $(R,\ S_1,\ Z)$;
Input : R - GPS Update Rate; S_1 - Last GPS Sample in PoA; Z - NFZ list.
Output: S_2 - Next GPS sample in PoA; $\text{Sig}(S_2, T^-)$ - Signature of S_2
while *true* **do**
  $S_2 \leftarrow$ `ReadGPS()`;
  $z \leftarrow$ `FindNearestZone`$(S_2,\ Z)$;
  $D_1 \leftarrow$ `Dist`$(S_1,\ z)$;
  $D_2 \leftarrow$ `Dist`$(S_2,\ z)$;
  if
   $S_2.t - S_1.t \le (D_1 + D_2)/v_{\max} < S_2.t - S_1.t + 2/R$
   then
    S_2, $\text{Sig}(S_2) \leftarrow$ `GetGPSAuth()`;
    return S_2, $\text{Sig}(S_2, T^-)$;
  else
    sleep$(1/R)$;
  end
end

V. Hardware and Implementation

A. Hardware Platform

We choose ARM Trustzone [8] as our secure hardware platform. Although Intel SGX [9] processors provide better performance in general, they do not emulate the resource limited computation environment of drone hardware. Similar to the secure enclaves in SGX, the TrustZone partitions the software and hardware into two worlds, a normal world and a secure world. The hardware logic ensures that the resources in the secure world is inaccessible from the normal world.

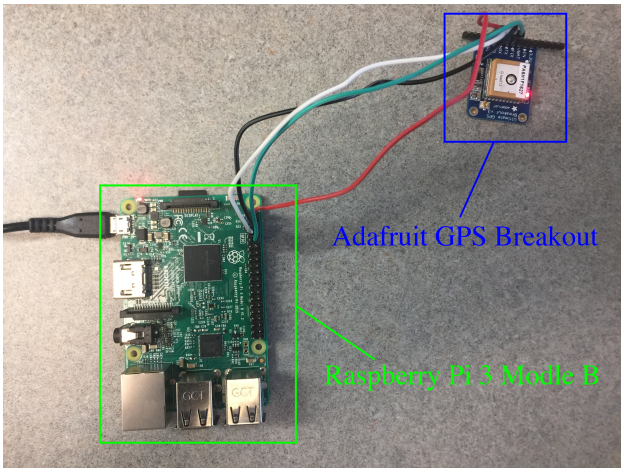

Fig. 5. Hardware Platform consists of a Raspberry Pi 3 Model B and Adafruit Ultimate GPS breakout.

Specifically, we implement a proof-of-concept prototype of AliDrone client on Raspberry Pi 3 Model B [22], which has a 1.2GHz 64-bit quad-core ARMv8 CPU that supports ARM TrustZone. Previous effort has shown feasibility of deploying a practical drone controller on Raspberry Pi [23].

We connect the Raspberry Pi with an Adafruit Ultimate GPS breakout [24] via GPIO ports as shown in Fig. 5. The sampling rate of the GPS receiver can be configured in the range of [1 Hz, 5Hz]. It outputs the GPS messages following the NMEA 0183 standard [25].

We acknowledge that a dishonest Drone Operator may instead connect a malicious GPS receiver or a programmable UART device to generate GPS messages by purpose. Consequence of such attack leads the Trusted Execution Environment to sign forged GPS messages and thus breaks the security guarantee of our system. Therefore, the manufacturers should consider an alternative hardware design by using embedded GPS chips to prevent this attack.

B. GPS Driver & GPS Sampler

The GPS Driver is implemented in kernel space of OP-TEE core. It maps the physical address of the GPIO RX port to a memory buffer. In particular, we are interested in the $GPRMC messages, which contains information including latitude, longitude, velocity and timestamps. The latest $GPRMC message is read from the buffer and parsed into (latitude, longitude, timestamp) tuple using an open-sourced library Libnmea [26]. An interface `GetGPS()` is exposed to the GPS Sampler, which returns the latest GPS tuple.

The GPS Sampler is implemented as a Trusted Application (TA) in non-privileged mode in the secure world. It uses the private sign key to authenticate the GPS tuples.

An interface `GetGPSAuth()` is provided to the Adapter. Once `GetGPSAuth()` is called, it reads the latest GPS tuple from the GPS Driver and then and signs the sample with the private sign key. Our implementation uses `TEE_ALG_RSASSA_PKCS1_V1_5_SHA1` algorithm to sign the GPS data.

C. Adapter

The Adapter is implemented as a daemon service in the user space of the normal world. We implement the adaptive sampling algorithm in the Adapter and gets the authenticated GPS tuples by calling `GetGPSAuth()` from the GPS Sampler. We use `RSAES_PKCS1_v1_5` algorithm to encrypt the GPS data with the public key of the Auditor and persist the ciphertext along with the signature in the local storage.

VI. Evaluation

A. Field Studies

In this section, we evaluate the AliDrone in two cases, each representing a specific pattern of the surrounding no-fly-zones.

1) Experimental Setup: We deploy the hardware described in section V-A and emulate the flight pattern of a drone using a vehicle. As the personal properties may be reserved as NFZs, it is reasonable to assume that the airspace upon roads and public areas like parks are available for commercial drone navigation. We emulate the GPS sampling of drones by driving the vehicle around a small county region. The maximum sampling rate of the GPS sensor was set to 5 Hz and the entire GPS traces including latitude, longitude and timestamps were recorded. The collected GPS data was replayed to the GPS Sampler to emulate the real-time GPS samples read from the GPS Driver interface.

We specified two sets of no-fly-zones into the AliDrone client. In the first case, we set a single NFZ with a large radius. This case represents large no-fly areas in the city or nearby critical infrastructures like airports and power plants. In the second case, we set multiple small and dense no-fly-zones along the route of the driving path. This case simulates the scenario where the drone flies through a residential area and it should not fly over any of the neighbors with no-fly-zone.

We compare the adaptive sampling with a baseline approach which we refer as "Fix Rate Sampling". Every time after a GPS data is sampled, the sampling thread will sleep for a period according to the sampling rate. Since the GPS hardware has an independent rate for updating the measurements, the sampler cannot always get the most updated GPS data immediately after it wakes up. Therefore, we let the sampler wait until the first measurement update for each time after it wakes up. As a result, the actual sampling rate is as fast as configured. For example, if the update rate of GPS hardware is 5Hz, five samples are produced in each second at $t = 0.0, 0.2, 0.4, 0.6,$ and 0.8s. If the sampler runs at 3Hz, it wakes up at $t = 0.0, 0.33,$ and 0.67s. Then the time that three samples are taken should be $t = 0.0, 0.4, 0.8$s.

2) Airport Scenario: FAA regulations forbid drone operations within 5 miles of any airport. In this scenario, we set an NFZ centered at an airport with a radius of 5 miles. The GPS trace starts about 30 feet outside the boundary of the NFZ. The vehicle drives away from the NFZ for about 3 miles in 12 minutes.

We set the sampling rate as 1Hz and keep track of the total number of GPS samples as well as the distance to the boundary. When the vehicle is close to the boundary, the sampling rates of fix rate sampling and adaptive sampling are similar. As the distance increases, the adaptive sampling requires fewer samples for a sufficient alibi. Comparing to the 649 samples collected by 1Hz fix rate sampling, the adaptive sampling uses only 14 GPS samples.

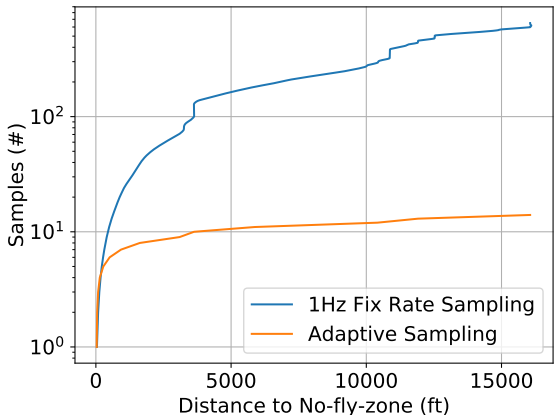

Fig. 6. In airport scenario, we keep track of the total number of GPS samples, and the distance between the vehicle and the boundary of the NFZ.

3) Residential Scenario: The residential areas are comprised of many small but dense NFZs. In this scenario, we drive the vehicle through a local county for about one mile. Fig. 7 shows the satellite view of the residential area and marks the driving route from location A to B. For purpose of anonymity, the names and labels are removed from the map. We use Google Maps to identify the houses along the driving route and mark each of them as an NFZ. Every NFZ is represented by a circle centers at a house with a radius of 20 feet. In total, 94 NFZs are identified in this area.

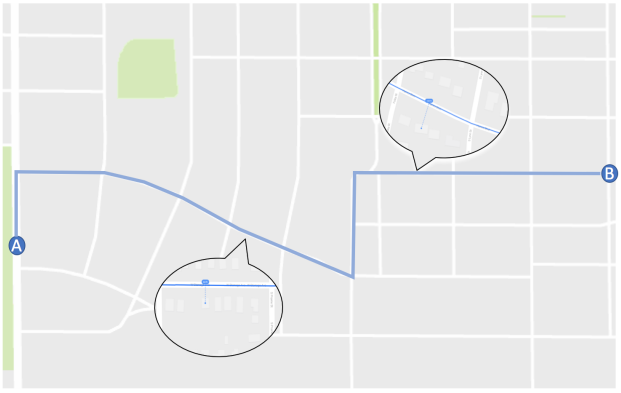

Fig. 7. Map and driving route of the residential area.

We are interested in three metrics in the residential scenario. **Distance to the nearest NFZ:** As the vehicle moves, its distances to the NFZs are changing. However, only the

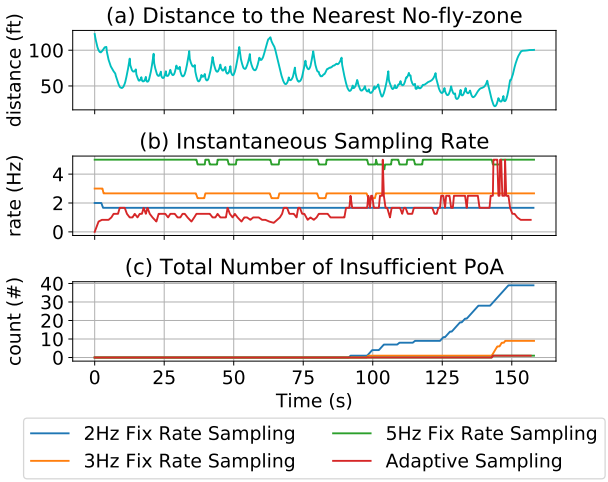

Fig. 8. We measure three metrics in the residential scenario: (a) distance to the nearest NFZ; (b) instantaneous sampling rate; (c) total number of insufficient Proof-of-Alibi.

nearest NFZ affects the sampling rate because a PoA proving alibi to the nearest NFZ is also sufficient for the other NFZs. The distance of the vehicle to the nearest NFZ is shown in Fig. 8-(a). Such distance indicates the density of the neighborhood. At the beginning, the distance is primarily within range 50 - 100 ft. When the vehicle enters a more dense area, the distance decreases to 20 - 70 ft. At the closest point, the vehicle is only 21 ft to the boundary of the nearest NFZ.

Instantaneous Sampling Rate: We compare the the instantaneous sampling rate of adaptive sampling to fix rate sampling with 2 Hz, 3 Hz and 5 Hz in Fig. 8-(b). Note that the sampler may wait for a small period time for the first GPS update, the actual sampling rate in Fix Rate Sampling can be lower than the settings. When the vehicle travels in the less dense area, the Adaptive Sampling uses a sampling rate lower than 2Hz. This saves the total number of GPS samples produced in PoA. As the vehicle enters the dense area, the adaptive algorithm pushes to higher sampling rate to preserve the sufficiency of PoA.

Total Number of Insufficient PoA: If the time between two continuous GPS samples is too long, the trace cannot provide sufficient PoA. For every continuous sample pair (x_i, y_i, t_i) and $(x_{i+1}, y_{i+1}, t_{i+1})$, we count the insufficient PoAs as follows:

$$\text{count } += \begin{cases} 1 & \text{if } \min_j(d_{i,j} + d_{i+1,j}) \leq v_{\max}(t_{i+1} - t_i), \\ 0 & \text{otherwise,} \end{cases}$$

where $d_{i,j}$ is the distance from the location of sample i to NFZ j.

Fig. 8-(c) demonstrates the total number of insufficient PoAs over time. In the first one and a half minutes, no insufficient PoA is spotted. As the vehicle drives into the dense area, the

fix rate sampling with 2Hz and 3Hz are unable to produce sufficient PoA. In total, 39 and 9 insufficient PoAs are counted in 2Hz and 3Hz Fix Rate Sampling.

Adaptive sampling achieves as few insufficient PoAs as fix rate sampling (5Hz). However, an insufficient PoA is identified at a time the vehicle is 25 ft to an NFZ. By further inspection of the GPS trace, we find that the GPS hardware misses an update when insufficient PoA takes place. This means that the maximum sampling rate drops from 5Hz to 2.5Hz at this point.

B. Benchmarks

In this section, we present the benchmarks of AliDrone by testing the processing and energy overhead in a controlled laboratory environment. The experimental platform of the benchmarks is Raspberry Pi 3 Model B, which has a 1.2 GHz 64-bit quad-core ARMv8 processor and a 1 GB LPDDR2-900 SDRAM memory. The CPU utilization and memory consumption of AliDrone are measured by running the GPS Sampler on a single core under a fixed sampling rate. The power consumption is derived from the power model presented by Kaup et al. [27]:

$$P_{\text{CPU}}(u) = 1.5778\text{W} + 0.181 \cdot u \cdot \text{W} \tag{4}$$

where u is the average CPU utilization ranging from 0 to 1.

We first run the GPS Sampler under a fixed sampling rate of 2 Hz, 3 Hz and 5 Hz for 5 minutes. We use `top` command to measure the CPU utilization and memory consumption once per second and take the average over all the measurements. Power consumption is computed by equation (4). Two encryption and sign key sizes (1024 and 2048 bits) are tested in the benchmarks. Then, we replay the GPS data collected from the two field studies and run the measurements again using the same settings.

Table II shows the benchmarks for CPU utilization, power consumption and memory consumption. Since the Raspberry Pi has four cores, the range of CPU utilization measurement is [0, 25%].

Key Size (bits)	Case	CPU (%)	Power (W)
1024	Fixed 2 Hz	2.17 ±0.05	1.5817
	Fixed 3 Hz	3.17 ±0.04	1.5835
	Fixed 5 Hz	5.59 ±0.06	1.5879
	Airport	0.024 ±0.160	1.5778
	Residential	1.567 ±0.827	1.5806
2048	Fixed 2 Hz	10.94 ±0.09	1.5976
	Fixed 3 Hz	16.81 ±0.10	1.6082
	Fixed 5 Hz	-	-
	Airport	0.122 ±0.810	1.5780
	Residential	-	-
Memory		3.27 MB (0.3%)	

Table II
CPU, Power and Memory Benchmarks

The benchmark results show that AliDrone only consumes a small amount of memory of about 0.3%, which suggests that it will not affect other memory intensive tasks. In terms of CPU utilization, AliDrone can support trustworthy GPS sampling with the maximum rate of 5 Hz using a short sign key (1024 bits). The computation overhead introduced by AliDrone is about 5.6% on average. In the case of large TEE sign key (2048 bits), AliDrone cannot keep up with the maximum sampling rate. This result implies that more efficient signature schemes are required to support higher GPS sampling rate.

The real-world benchmarks demonstrate that the adaptive sampling mechanism can further reduce the processing overhead. Running AliDrone in a dense residential county using a 1024-bit sign key only costs an average of 1.5% CPU cycles. Again the measurement under 2048-bit sign key is not presented because of the large overhead of computing asymmetric signatures.

VII. Discussion

In this section, we discuss the limitations and future extensions of AliDrone.

A. Limitations

1) Cryptographic Operations with Long Keys: As mentioned in section VI-B, high sampling rate is needed when the drone is close to the no-fly-zones. However, the hardware may be unable to keep up with the sampling rate if the keys are long. As the asymmetric encryption and signature operations are costly in the resource limited hardware platform, we may consider the following options.

a) Symmetric cryptography: A drone may setup ephemeral symmetric keys with the Auditor every time before it starts a flight. Such keys can be used to encrypt and sign the GPS data. However, the sign key must be inaccessible by the Drone Operator. Thus, a key exchange protocol is needed between the Drone TEE and the Auditor.

b) Sign all traces at once: We may consider the option that caches the GPS samples in the secure memory and sign the whole trace at once. This is feasible because the flight time of drones are usually no more than 30 minutes and the size of each GPS sample is small.

2) GPS Spoofing Attacks: Our solution does not consider GPS spoofing attacks, in which attackers send incorrect GPS signals to manipulate the GPS receiver. It is even more challenging to mitigate such attacks in the context that the attacker is the owner of the drone.

A potential solution can be developed by embedding the GPS spoofing detector into the secure world [18]–[20]. If the hardware is running in a suspicious environment, the GPS Sampler can decline to provide authenticity services.

B. Future Extensions

1) 3D Physical Model: We briefly demonstrate how to extend AliDrone to include altitude information using a 3-dimensional physical model.

The GPS sample can be modified as a 4-tuple $S = (lat, lon, alt, t)$, where alt represents the altitude of the drone. Similarly, the altitude dimension should be added to the NFZ specification, i.e., $z = (lat, lon, alt, r)$ can be interpreted as a cylinder no-fly-region.

Given two consecutive GPS samples $S'_1 = (x_1, y_1, z_1, t_1)$ and $S'_2 = (x_2, y_2, z_2, t_2)$, the possible traveling range can be described as an *ellipsoid*

$$\mathcal{E}'(S'_1, S'_2) = \{(x, y, z) \mid d_1 + d_2 \leq v_{\max}(t_2 - t_1)\},$$

where $d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$. Hence, these two GPS samples can prove alibi to an NFZ $z' = (x_0, y_0, z_0, r_0)$ if and only if the ellipsoid does not intersect with the cylinder, i.e.,

$$\mathcal{E}'(S'_1, S'_2) \cap z' = \varnothing.$$

2) Arbitrary No-fly Zones: The design of proof-of-alibi assumes that all no-fly zones are circular. In real world applications, the zone owners may want to register a no-fly zone with a non-circular shape. We show how to extend AliDrone to adapt with arbitrary shaped NFZs.

At the NFZ registration phase, the zone owner can describe the NFZ by a polygon with n vertices. The auditor will first find a smallest circle that covers all the vertices, and use this circle to represent the NFZ. This problem is well known as the *smallest circle problem* and can be solved in linear time [28]. The computation of each NFZ only happens once at the registration phase. Therefore, the computation cost for the auditor can be negligible.

3) Privacy-preserving Verification: The design of AliDrone relies on a trusted Auditor to verify the PoAs. However, a dishonest Auditor may take advantage of AliDrone to track drone using the geo-location information in PoAs. Such an adversary may be an employee or insider of FAA who tends to leak the GPS traces of commercial drones and gains profit from this activity.

AliDrone can be extended by assuming the presence of an *honest-but-curious* Auditor. The privacy-preserving extension aims to prevent the Auditor from learning the entire GPS trajectory from PoAs while still allows it to conclude on a boolean value whether the drone violates NFZ compliance.

A potential solution can be developed using one-time encryption scheme [29]. Drone Operators can use one-time keys to encrypt each GPS sample S_i in the PoA, and uploads the encrypted PoA to the Auditor. When a Zone Owner spots a drone in the NFZ, she reports the potential violation to the Auditor by sending a message including her zone id, the drone's id and time of incident. By receiving the accusation, the Drone Operator can reveal the two corresponding GPS samples in the PoA by sending the two one-time encryption keys. In this way, the Auditor can only learn the partial GPS trajectory of the drone.

VIII. Related Work

A. Drone Privacy

Privacy is one of the major concerns about the pervasive deployment of UAVs. Nevertheless, among cybersecurity, privacy and public safety issues [30], the previous research on drone privacy was limited to regulations [31]–[33]. The most promising approach suggested by Cavoukian [34] was to apply Privacy by Design (PbD) principle to drone technologies. However, even though the drone system is complied with PbD, an authorized drone operator can always attach a small camera to the drone and covertly capture surveillance videos.

To the best of our knowledge, our work is the first technical solution enabling drones to present proof of privacy compliance.

B. Trusted Execution Environment

Trusted Execution Environment (TEE) technology has received excessive research interest in recent years. Intel software guard extension (SGX) [35], ARM TrustZone [8], [36] and virtual TEE solution [37] made it possible to secure sensitive code or data even if the system is comprised with root access.

Utilizing TEE will allow us to build systems with stronger security privileges [38], [39]. Specifically, Schuster et al. [40] presented a verifiable Hadoop system which keeps code and data secret even if the machines in a data center are compromised. Zhang et al. [41] designed an efficient two-factor authentication scheme on ARM TrustZone and achieved comparable security assurance to hardware token based solution. Liu and Srivastava [42] used ARM TrustZone to protect essential

C. Location Forgery

An abundant amount of work focused on GPS spoofing, where the attacker sends spoof signals to confuse victim's GPS device [17], [43], [44]. In contrast, very few literatures were to deal with the issue that users forge the location information based on correct GPS readings.

Li et al. [45] presented a defense against "localtion cheating attack" by issuing location proof from the nearby WiFi access points. Oh et al. [46] allowed nearby mobile devices to cooperate and to unauthorize a forged location generated by attackers. However, these approaches are not suitable for detecting location forgery in drones. First, there is no fixed object like WiFi access points in the air so as to help with the location verification. Second, while commercial drones are flying in the air, they can hardly form a short range wireless network because high density of drones can lead to higher probability of collision.

D. Drone Routing

Although we do not focus on the routing problem in this paper, this class of literature is complementary to our work, and can be used to optimize the Proof-of-Alibi. The routing algorithms are specifically designed depending on the task that the drone is part of [47], [48].

Our approach to generate PoA is similar to the one used in [49], in which the network routing system can generate proof that the route does not enter certain prohibited areas specified by the sender of packets.

IX. Conclusion

In this work, we have considered the privacy issue in regard to drones. In specific, we solve the challenging task of determining whether a drone has entered NFZs in small and dense areas. We design a lightweight Proof-of-Alibi system to enable drones to show NFZ compliance. Consequently, our system facilitates trustworthy auditing of drone privacy compliance.

Acknowledgement

This work was supported in part by XXX CNS grants XXX, and XXX. The views expressed are those of the authors only.

References

[1] (2016, Oct.) Amazon air prime. https://goo.gl/3NHNre.
[2] (2016, Oct.) How drones are being used in 2016. https://goo.gl/ZbQirw.
[3] P. J. Hiltner, "Drones are coming: Use of unmanned aerial vehicles for police surveillance and its fourth amendment implications, the," *Wake Forest JL & Pol'y*, vol. 3, p. 397, 2013.
[4] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "Ar-drone as a platform for robotic research and education," in *International Conference on Research and Education in Robotics*. Springer, 2011, pp. 172–186.
[5] (2016, Oct.) FAA summary of small unmanned aircraft rule (part 107). https://goo.gl/JbpgST.
[6] (2016, Oct.) FAA B4UFLY. https://goo.gl/EdxuJ9.
[7] (2017, Oct.) Codrone. https://www.thisiswhyimbroke.com/worlds-first-programmable-drone/.
[8] (2016, Oct.) Arm tee reference documentation. https://goo.gl/TGq7Kg.
[9] Intel. (2017, May) Intel software guard extensions (sgx) sdk. https://goo.gl/vp3Xm3.
[10] (2017, May) Op-tee trusted application. https://goo.gl/72ebW1.
[11] J. Daniels, "As Sand Fire rages, feds turn up heat in fight against drones interfering in wildfires," https://goo.gl/NBECb5, Jul. 2016.
[12] B. Coppens, I. Verbauwhede, K. De Bosschere, and B. De Sutter, "Practical mitigations for timing-based side-channel attacks on modern x86 processors," in *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE, 2009, pp. 45–60.
[13] M.-W. Shih, S. Lee, T. Kim, and M. Peinado, "T-sgx: Eradicating controlled-channel attacks against enclave programs," in *ISOC Network and Distributed System Security Symposium*, 2017.
[14] N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza, "Asyncshock: Exploiting synchronisation bugs in intel sgx enclaves," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 440–457.
[15] S. Peterson and P. Faramarzi, "Exclusive: Iran hijacked US drone, says Iranian engineer," https://goo.gl/3gK56T, Dec. 2011.
[16] J. Saarinen, "Students hijack luxury yacht with GPS spoofing," https://goo.gl/BvXi61, Jul. 2013.
[17] K. D. Wesson, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "An evaluation of the vestigial signal defense for civil gps anti-spoofing," in *Proceedings of the ION GNSS Meeting*, 2011.
[18] M. L. Psiaki, B. W. O'Hanlon, J. A. Bhatti, D. P. Shepard, and T. E. Humphreys, "Gps spoofing detection via dual-receiver correlation of military signals," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2250–2267, 2013.
[19] B. W. O'Hanlon, M. L. Psiaki, J. A. Bhatti, D. P. Shepard, and T. E. Humphreys, "Real-time gps spoofing detection via correlation of encrypted signals," *Navigation*, vol. 60, no. 4, pp. 267–278, 2013.
[20] J. Wang, W. Tu, L. C. Hui, S. Yiu, and E. K. Wang, "Detecting time synchronization attacks in cyber-physical systems with machine learning techniques," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2246–2251.
[21] I. Leveson, "Benefits of the new gps civil signal," *Inside GNSS*, vol. 1, no. 5, pp. 42–47, 2006.
[22] (2017, May) Raspberry pi 3 model b. https://goo.gl/tHUidL.
[23] (2017, May) The drone pi. https://goo.gl/vzqajc.
[24] (2017, May) Adafruit ultimate gps breakout. https://goo.gl/MSMXiy.
[25] R. Langley, "Nmea 0183: A gps receiver," *GPS world*, 1995.
[26] (2017, May) Libnmea: C library for parsing nmea 0183 sentences. https://goo.gl/An9Xq6.
[27] F. Kaup, P. Gottschling, and D. Hausheer, "Powerpi: Measuring and modeling the power consumption of the raspberry pi," in *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*. IEEE, 2014, pp. 236–243.
[28] N. Megiddo, "Linear-time algorithms for linear programming in r^3 and related problems," *SIAM journal on computing*, vol. 12, no. 4, pp. 759–776, 1983.
[29] M. Abdalla, O. Chevassut, and D. Pointcheval, "One-time verifier-based encrypted key exchange." in *Public Key Cryptography*, vol. 3386. Springer, 2005, pp. 47–74.
[30] E. Vattapparamban, İ. Güvenç, A. İ. Yurekli, K. Akkaya, and S. Uluağaç, "Drones for smart cities: Issues in cybersecurity, privacy, and public safety," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International*. IEEE, 2016, pp. 216–221.
[31] R. Calo, "The drone as privacy catalyst," *Stanford Law Review Online*, vol. 64, pp. 29–33, 2011.
[32] A. Harrington, "Who controls the drones?[regulation unmanned aircraft]," *Engineering & Technology*, vol. 10, no. 2, pp. 80–83, 2015.
[33] B. Jenkins, "Watching the watchmen: Drone privacy and the need for oversight," *Ky. LJ*, vol. 102, p. 161, 2013.
[34] A. Cavoukian, *Privacy and drones: Unmanned aerial vehicles*. Information and Privacy Commissioner of Ontario, Canada Ontario, Canada, 2012.
[35] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13, 2013.
[36] Lenaro. (2016, Oct.) Op-tee. https://goo.gl/53ZVmy.
[37] B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, "Open-tee–an open virtual trusted execution environment," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1. IEEE, 2015, pp. 400–407.
[38] J. Winter, "Experimenting with arm trustzone–or: How i met friendly piece of trusted hardware," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2012, pp. 1161–1166.
[39] M. Pirker and D. Slamanig, "A framework for privacy-preserving mobile payment on security enhanced arm trustzone platforms," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2012, pp. 1155–1160.
[40] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "Vc3: trustworthy data analytics in the cloud using sgx," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 38–54.
[41] Y. Zhang, S. Zhao, Y. Qin, B. Yang, and D. Feng, "Trusttokenf: A generic security framework for mobile two-factor authentication using trustzone," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1. IEEE, 2015, pp. 41–48.
[42] R. Liu and M. Srivastava, "Protc: Protecting drone's peripherals through arm trustzone," in *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*. ACM, 2017, pp. 1–6.
[43] X.-j. Cheng, K.-j. Cao, J.-n. Xu, and B. Li, "Analysis on forgery patterns for gps civil spoofing signals," in *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International Conference on*. IEEE, 2009, pp. 353–356.
[44] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 75–86.
[45] Y. Li, L. Zhou, H. Zhu, and L. Sun, "Privacy-preserving location proof for securing large-scale database-driven cognitive radio networks," 2012.
[46] S. Oh, T. Vu, M. Gruteser, and S. Banerjee, "Phantom: Physical layer cooperation for location privacy protection," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 3061–3065.
[47] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," 2016.
[48] T. R. Jorris, "Common aero vehicle autonomous reentry trajectory optimization satisfying waypoint and no-fly zone constraints," DTIC Document, Tech. Rep., 2007.
[49] D. Levin, Y. Lee, L. Valenta, Z. Li, V. Lai, C. Lumezanu, N. Spring, and B. Bhattacharjee, "Alibi routing," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 611–624.