LiveBox: A Self-Adaptive Forensic-Ready Service for Drones

How to cite:

For guidance on citations see FAQs.

© 2019 IEEE

https://creativecommons.org/licenses/by/4.0/

Version: Version of Record

Link(s) to article on publisher’s website:
http://dx.doi.org/doi:10.1109/ACCESS.2019.2942033

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.
LiveBox: A Self-Adaptive Forensic-Ready Service for Drones

YIJUN YU1, (Member, IEEE), DANNY BARTHAUD1, BLAINE A. PRICE1, AROSHA K. BANDARA1, ANDREA ZISMAN1, AND BASHAR NUSEIBEH1,2

1School of Computing and Communications, The Open University, Milton Keynes MK7 6AA, U.K.
2Lero—The Irish Software Research Centre, University of Limerick, V94 NYD3 Limerick, Ireland

Corresponding author: Yijun Yu (y.yu@open.ac.uk)

This work was supported in part by the Engineering and Physics Sciences Research Council (EPSRC), in part by the Science Foundation Ireland (SFI), in part by the EU Single European Sky ATM Research Consortium Knowledge Transfer Network (SESAR KTN), in part by the National Endowment for Science, Technology and the Arts (NESTA), and in part by the Sao Paulo Research Foundation (FAPESP).

ABSTRACT

Unmanned Aerial Vehicles (UAVs), or drones, are increasingly expected to operate in spaces populated by humans while avoiding injury to people or damaging property. However, incidents and accidents can, and increasingly do, happen. Traditional investigations of aircraft incidents require on-board flight data recorders (FDRs); however, these physical FDRs only work if the drone can be recovered. A further complication is that physical FDRs are too heavy to mount on light drones, hence not suitable for forensic digital investigations of drone flights. In this paper, we propose a self-adaptive software architecture, LiveBox, to make drones both forensic-ready and regulation compliant. We studied the feasibility of using distributed technologies for implementing the LiveBox reference architecture. In particular, we found that updates and queries of drone flight data and constraints can be treated as transactions using decentralised ledger technology (DLT), rather than a generic time-series database, to satisfy forensic tamper-proof requirements. However, DLTs such as Ethereum, have limits on throughput (i.e. transactions-per-second), making it harder to achieve regulation-compliance at runtime. To overcome this limitation, we present a self-adaptive reporting algorithm to dynamically reduce the precision of flight data without sacrificing the accuracy of runtime verification. Using a real-life scenario of drone delivery, we show that our proposed algorithm achieves a 46% reduction in bandwidth without losing accuracy in satisfying both tamper-proof and regulation-compliant requirements.

INDEX TERMS

Unmanned aerial vehicles (Drones), software engineering, self-adaptive systems, forensic readiness, flight data recorders, simulators, unmanned traffic management.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), or drones, are increasingly expected to operate in spaces populated by humans, while avoiding injury to people and damage to property. In the USA alone, the number of drones is predicted to reach 7.5 million by 2020, and with global companies like Amazon testing them to deliver goods to households, the (UK) Civil Aviation Authority (CAA) has begun to identify issues with regulatory compliance.

Drones must also respect the privacy of people on the ground, and avoid restricted spaces such as those occupied by civilian aircraft or sensitive sites. However, incidents and accidents can, and increasingly do, happen, e.g. on 16 October 2017.1 According to the UK Airprox Board [1], more near-miss incidents are caused by drones than by aircraft, leading to a “new law for UK users to sit safety tests”.2

The traditional method for investigating an aircraft incident is to examine the flight data recorders (FDRs). For drones, this is challenging because the FDRs are too heavy and do not provide real-time data. FDRs also require physical access to the drone, which may not be available when carrying out an investigation into a drone which has flown away.

The following real world scenario highlights several of our research challenges.

Scenario: Transport for London (TfL) is considering the deployment of drones to deliver medical assets such as blood

1D. Lee, ‘Drone collides with a commercial aeroplane’, BBC News
2UK drone users to sit safety tests under new law,http://www.bbc.co.uk/news/uk-42126150
and organs between hospitals. In the past, blue-light vehicles were used by the London Ambulance Service to deal with such medical emergencies. Blood and organs need to be delivered efficiently and reliably, which is not always viable in a traffic-congested mega city. Therefore, motor-bikes are often used instead. Cost aside, motor-bikes are noisy and one of the causes of traffic accidents. The use of drones for this purpose has therefore been proposed. However, when drone-related incidents happen, they must be analysed quickly, including checks for their compliance to the regulations of the airspace they navigated.

The above scenario illustrates the challenges in satisfying two key requirements:

- **Forensic-soundness**: Digital evidence collected must be tamper-proof by design, admissible in court, and accurate enough to avoid errors in forensic investigations;
- **Regulation-compliance**: The evidence must be capable of being used to demonstrate whether or not a drone has violated current regulations.

To address these challenges, in this paper we propose a distributed software architecture called ‘LiveBox’. In addition to satisfying the above two core forensic-readiness requirements, several non-functional requirements are also critical. The LiveBox services must be scalable to handle large volumes of sensor data that will be collected, as well as resilient in the face of accidental, environmental or intentional interference in collection and handling of the data needed to investigate incidents. The integrity of the data and subsequent analysis must be robust, even in the absence of mutual trust between the stakeholders involved.

To provide an uninterrupted forensic-ready LiveBox service, self-adaptation mechanisms need to perform the activities that form a MAPE-K feedback loop: monitoring (M), analysis (A), planning (P), and execution (E), around situation-awareness knowledge (K). This knowledge is part of the forensic evidence collected.

The novel contribution of LiveBox in this paper is the representation of flight data of drones as tamper-proof blockchain ledgers, and the verification of the consensus of these flight data through a self-adaptive reporting algorithm using locality-sensitive hashes and making situation-aware decisions on demand. Overall, a smart contract-based blockchain representation of flight data enables recovery of the knowledge about drone flight records as tamper-proof evidence; a self-adaptive reporting algorithm on board monitors the distance between the drone and the zone boundary, analyses an approximate geolocation to report with minimal digits, plans for skipping certain reports when current situation can be inferred, and executes minimal reporting instructions without sacrificing the two core forensic-readiness requirements.

The rest of the paper is structured as follows. Section II presents a reference architecture of LiveBox to address the two core requirements: forensic soundness and regulation compliance. Section III illustrates the basic research problem through some pilot studies and indicates why a self-adaptive solution is needed. Section IV proposes and evaluates a blockchain smart contract and a self-adaptive reporting algorithm to address the technical challenges in achieving both requirements for real-life drone forensic readiness scenarios. Section V discusses related work, and Section VI summarises our findings.

## II. LIVEBOX REFERENCE ARCHITECTURE

Our proposed reference software architecture for LiveBox is shown in Figure 1. It comprises three kinds of software services:

![LiveBox architecture](Figure 1)

(i) **On-drone LiveBox Services.** Captures flight data from various physical sensors and receivers with situation awareness of other drones through automatic dependent surveillance-broadcast (ADS-B) [2]. This live data is cached locally and communicated as streams of time-critical data to the LiveBox cloud service, preserving privacy when possible;

(ii) **The LiveBox Cloud Service.** Allows stakeholders to specify no-fly zones as constraints, against which drones are checked during forensic investigations. The trusted storage solution uses Distributed Ledger Technology (DLT) as a scalable, tamper-proof, and reliable way of handling forensic investigation or regulation compliance queries;

(iii) **LiveBox Ground Service.** Uses additional sensors of mobile or fixed ground stations to capture signals of drones, whether or not they have on-drone LiveBox services. This flight data is also communicated to the LiveBox Cloud Service to cross-validate the behaviour of drones. When a drone violates regulations due to emerging failure conditions (e.g., changing weather, lost connection, or temporary no-fly zones), a notification is sent to a LiveBox compatible drone controller for intervention and also used by other LiveBox services to adaptively enhance forensic data collection.

To support all such communications requires protocols to support forensic analysis of live flight data, on or off the drones. Such protocols, however, are beyond the scope of this paper.

Focusing on the software engineering characteristics of LiveBox, two research challenges need to be addressed:
RC1 Collecting and storing drone flight data, in real-time, through a distributed mechanism, enabling forensically sound investigations;

RC2 Live forensic analysis of flight data to check compliance with regulations such as ‘no-fly’, ‘flight corridor’, or ‘privacy zone’ zones, or non-compliance with the ‘line of sight’ rules for drone operations;

III. FROM LIVE BLACKBOX TO LIVEBOX

Since LiveBox is a complex system that extends beyond software services, it is infeasible for us to demonstrate a holistic solution (including hardware) yet. However, with respect to the key properties and requirements, we can report preliminary findings through pilot experiments that illustrate the feasibility of our approach.

Live blackbox [3] was a pilot study we conducted to check the feasibility of achieving five tracking and verification requirements with respect to logging aircraft data, namely, Tracking, Prediction, Scalability, Verification, and Liveness. Tracking concerns collecting flight data from aircraft, Prediction concerns sufficient accuracy in knowing the next location of aircraft, Scalability concerns doing these at the global scale with thousands of aircraft, Verification concerns the capability to tell whether an aircraft is within normal boundary or not, and Liveness concerns the timely, live update of such flight data and the services [4] to forensic investigators. Furthermore, to accommodate the ephemeral nature of flights, we also considered Snap forensics requirements to bound the liveness by expiry timestamps [5].

In addition to these findings, it is worth emphasising that drone forensics requires the LiveBox data to be tamper-proof, while our preliminary Live Blackbox study assumed it from properties of blackbox hardware, without any additional software measures in place to enforce it.

As drones are different from aircraft in terms of limited battery capacity (most drones can fly no longer than 30 minutes per session), limited communication distances (usually ADS-B is not used except for very high-end drones), higher update frequency (usually in seconds rather than in minutes), etc., one cannot take for granted that a Live Blackbox solution is sufficient for the proposed LiveBox. In the following, we describe a number of pilot experiments we conducted to test how feasible it is to implement each LiveBox sub-service proposed. Later on, we will also discuss the limitations in these pilot studies to achieve the vision of self-adaptive live boxes.

A. REAL-WORLD PROBLEMS AND PILOT STUDIES

Many high profile ‘near-miss’ incidents have been logged by the UK Airprox Board (see e.g. [1]). These are qualitative records when pilots or someone on board an aircraft report to an authority (e.g., CAA) about an object, such as a bird or an unidentified drone, that flies by dangerously. Despite the tightening safety standards, one can see from the trends over the last 20 years that the number of incidents has started to climb in recent years (see Figure 2a), and the growth is, unfortunately, accelerating for drone-related incidents (Figure 2b). These phenomena are an international concern recognised, for example, by the FAA in the USA and the ICAO in Canada – in a recent incident on 17 October 2017, an 8-passenger aircraft in Quebec was actually hit by a drone.3

In addition, regulator constraints with respect to aircraft safety have been imposed on the drone pilots, through the development of an official app ‘Drone Assist’ by NATS and Altitude Angel. Figure 3 illustrates a location-based plot of the no-fly zones, with different colours indicating the risk levels if a violation occurs. As one can see, geometrical boundaries of the no-fly zone regions are not always circular. Their shape can be irregular in the case of, for example, prisons and other points of interests. A pilot is required to restrict their flights to areas outside these boundaries.

When it comes to a metropolitan city such as London, the no-fly zones can be congested and overlapping, as shown in Figure 4. However, that does not mean it is impossible to fly a drone there, because not all no-fly zones are in force at all times for everyone. For example, the London Metropolitan Police has been allowed to send surveillance drones to sites like London bridge at night, or any time when buildings such as Grenfell Tower in London4 were on fire when residents inside needed to be rescued.

3http://www.bbc.co.uk/news/technology-41635518
Flight data can also be misreported, even though they might have been captured automatically. Figure 5a shows the status of a Parrot Bebop2 drone in terms of connectivity, GPS availability, battery levels, and crash status. In this view, several ‘crash’ events are in fact just normal landings in a grass field. Figure 5b gives a satellite view of the flight path, where different segments of the path are plotted along with the two nodes indicating the “Start” and “End” of the journey. It is worth noting that the GPS locations, when available, do not necessarily require the storage of satellite images because they can be pinned to the satellite images later. Figure 5c presents the plots of individual parameters including altitude, speed and battery percentages, all aligned with the time stamps. As long as such data cross-validate, their internal integrity (with respect to physical law compliance) could be checked using differential equations. However, storing the flight data of drones on the controller (mobile phones in this case) is no guarantee of tamper-proof integrity. It is possible to tamper with such data by replacing the files on the devices once attackers know where the database files are accessible.

B. ON-DRONE DATA COLLECTION AND INSTRUMENTATION

To check whether drones without GPS sensors (e.g., Parrot AR Drone 2.0) could still have accessible flight data, we used an open-source Image Forensic tool EXIF\(^5\) to extract detailed data videos stored on the drone in MPEG format. Table 1 shows a snapshot of such records below (the floating point numbers are already rounded up to 3 digits, where the exact number is much more precise). The first column is the percentage of battery, the second is a number indicating the discrete status (0 = unknown, 1 = initiated, 2 = landed, 3/7 = flying, 4 = hovering, 5/6 = taking off, 8 = landing, 9 = looping, etc.), the third, fourth, fifth columns are rotation degrees, in terms of Yaw, Roll, and Pitch and the last column is the altitude in metres.

When a drone is not flying, commercial tools such as Oxygen Drone Forensics kit could also be used to recover flight data from the memory chips and SD-cards. However, when a drone is flying and one would like to obtain live streamed data, it is possible to carry a mobile SIM card

\(^5\)https://www.sno.phy.queensu.ca/ phil/exiftool
FIGURE 5. Different types of flight data recorded by FreeFlight.

with telecommunication chips with the drone, to track their movement using a LiveBox cloud service.

C. LIVEBOX CLOUD SERVICE

There are many ways to implement the LiveBox cloud service; e.g., using a centralised server on the Internet, or distributed ground devices to pick up the signals on the volunteering basis. Both the OpenSky and FlightRadar24 networks record air traffic data with ADS-B radar transponders if they communicate with a networked node.

However, it has been reported that OpenSky and FlightRadar24 do not capture the same datasets, and it is possible to do an insertion attack to report fake flight data into the network. Currently, both OpenSky and FlightRadar24 services do not have a way to distinguish the truthful flight data from the false ones.

Blockchains are known to be tamper-proof, in the sense that to modify the consensus reached previously, it would require more than 50% of all the mining power, which is almost impossible in reality when there are a large number of decentralised nodes.

Public ledgers, such as Bitcoin, offer a possible storage medium for the flight data. However, more specialist blockchain technology capable of running smart contracts; e.g., Ethereum and Neo, are better suited to both data storage and retrieval.

IV. SELF-ADAPTIVE FORENSIC READINESS

In this section, we propose a feasible solution to provide forensic ready LiveBox services by evaluating alternatives in distributed technology and self-adaptation.

A. TAMPER-PROOF TO PROVIDE DATA INTEGRITY

If the data reported could be modified or tampered with, the intelligence would not be treated as forensically sound evidence, as indicated by the research challenge (RC1). Therefore, we formulate the following research question (RQ) to evaluate various flight data storage mechanisms against the tamper-proof property.

RQ1 When the flight data are live streamed, can they be tampered with to change the verification results?

In experiments we compared two mechanisms for live data streaming including time-series distributed databases such as InfluxDB [6], which is specialised to record Internet of Things (IoT) data for efficient and scalable handling of time-series queries; and blockchain technology such as Ethereum [7]–[9], which uses a distributed ledger to store transactions.

For the sake of simplicity, we assume in our evaluation that the reported locations are the same as those detected on board the drones.

To evaluate a mechanism by the tamper-proof property, we launch two kinds of attacks against it:

- inject false locations into the data records and remove true locations;
- inject false zone constraints into the data records and remove true zone constraints;

Only if both kinds of data injection attack fail, can the mechanism be used for our forensic readiness solution.

For InfluxDB, by inserting false locations post-mortem, that is, inserting the record of a geolocation with a time stamp older than a previously recorded event, we found that tampering is possible. Even worse, a query of the flight path after such an insertion attack would show the inserted geolocation as if it had occurred earlier. Similarly, the zone constraints key-value pairs can also be inserted post-mortem, while subsequent queries cannot distinguish them from the genuine transactions. Even though InfluxDB is highly efficient at querying and updating time-series data (in our batch experiment the transaction rate peaked above 10000 per second), we found it an infeasible solution to satisfy the tamper-proof requirement and the validity of the recorded data could be questioned if one cannot eliminate the possibility of injection attacks. In contrast, the rate of transaction
with Ethereum is reported 15 per second,\(^6\) which is the cost paid for higher-level integrity. However, one could package multiple FDR into one transaction to alleviate the situation.

We have also launched injection attacks against our Ethereum smart contract deployed to the Ethereum test network Rinkeby. Since the smart contract transaction timestamps are recorded as part of the blockchain ledger using crypto hashes, an injected transaction would raise an inconsistency between the time-stamp of drone position and the time-stamp of the blockchain transaction. Therefore, a transaction cannot be inserted post-mortem, eliminating the possibility of either inserting (resp. removing) a false (resp. true) flight data record or zone constraint record.

Under normal conditions the public Ethereum network is capable of verifying a transaction within 15 seconds, given that there is sufficient incentive for the miners (or signers in Ethereum speak) to include the transaction into the current block. As shown in Figure 4 we have deployed a smart contract on the Ethereum Rinkeby test network, where transactions have been executed to record the flight data.\(^1\)

Since Ethereum handles incoming transactions as a pipeline, the overall throughput can be larger than 15 seconds per transaction because the incoming transactions in a queue are periodically processed into the block in a batch, even though an individual transaction may take up to 15 seconds to be verified. In the worst case scenario, high frequency updates for an individual drone, would result in queuing up unverified transactions for that drone. Whilst a trust-less public blockchain may not meet such needs of high frequency update, there are alternative private blockchain solutions one can utilise. In this work, we assume that the update frequency per drone can be accommodated by Ethereum-like blockchains using a self-adaptive reporting mechanism described in the next subsection.

Using smart contracts (written in the Solidity programming language) in Ethereum, we have achieved a preliminary implementation to store the data on Table 1 into the distributed ledger.

While blockchain ledgers were initially designed for financial transactions, smart contracts have enabled them to be used for a wide range of applications. We are making use of them for recording the movement of drones in the cyber-physical airspace. By ‘cyber-physical airspace’, we mean that the spatial locations may not be the ‘exact places’ to which one can associate interpretable semantics [10]. For example, the safety implication is significantly different if the reported GPS location is near to airports or prisons, rather than an open field where no harm could be done to aircraft or people. Furthermore, the same physical location may have different connotations depending on the misuse cases of drones (e.g., near-misses or drug deliveries). The key point here is to settle the analogous ‘double-spending’ disputes in the terminology of distributed ledgers, where blockchains are designed to verify this by proof-of-work or proof-of-stake consensus protocols. In the case of near-miss incidents, the same drone cannot appear in more than one partitioned airspaces at any given time; in the case of drug deliveries, airspace around a prison may be treated differently during daylight than during the night.

\(^6\)https://www.coindesk.com/information/will-ethereum-scale
B. SELF-ADAPTIVE REPORTING ALGORITHM

Due to inevitable communication delays of the LiveBox verification, a key to achieving self-adaptation is to build a prediction model based on the previously built ones. This is articulated as the following research question to address the regulation-compliance research challenge (RC2) timely.

**RQ2** How accurate do geolocations need to be to tell precisely whether a drone is inside or outside a zone?

Drones on the move in physical space have a certain degree of continuity. Let \( \tau \) be the time interval between geolocation, and the drone is located at \( L(t) \) at the time \( t \) where \( L \) can be some dimension of the geolocation such as latitude, longitude, or altitude. The new location \( L(t + \tau) \) at the time \( t + \tau \) should satisfy \( (L(t + \tau) - L(t))/\tau \leq v \), where \( v \) is the maximum speed of the drone, there exists a maximum distance that a drone can travel away away from its current location \( L(t) \) for a known time interval.

\[
L(t + \tau) = L(t) + v \frac{(t + \tau) - t}{\tau} = L(t) + v \frac{\tau}{\tau} = L(t) + v \tau
\]

The radius of the reported circle is adaptable so that it is less likely to come to a wrong conclusion from the reports. Similarly, when the marker is inside the zone, one can adapt the report precision so that it remains accurate with respect to whether the drone is inside or outside of a boundary.

The second adaptable parameter is the frequency of reporting. One has to be careful though because by adjusting the time intervals between two consecutive reports, it is possible to deduce incorrect conclusions. As shown in Figure 7, the markers not reported may contain a violation to the flight corridor. By counting how many markers may be wrongly reported, it is possible to calculate the maximum possible interval that maintains compliance to the flight corridor.

Returning to the TFL drone flight scenario, increasing the intervals from reporting every 30 seconds to every 2 minutes, fewer way-points will be communicated, see Figure 10. Even when all of them are within the flight corridor, it is still not necessarily true that the entire journey has been compliant to the zone constraints.

To develop a self-adaptive algorithm to adjust the reporting accuracy, we created a feedback loop to rounding the locations to a tolerable precision, depending on their distances to the boundary of no-fly/flight corridor zones \( Z \).

Our basic concept of self-adaptive reporting is illustrated in Figure 7. Given an irregularly-shaped flight corridor through an area that is completely enclosed by a “no fly zone”, we first check whether a drone at a marked geolocation at a given (numbered) time stamp is inside or outside the flight corridor. We report an approximate location within an error circle around the exact location so that one can always tell from the reported location if the drone is inside or outside the flight corridor.

The level of precision is adaptively selected for each geolocation based on its distance from the flight corridor boundary to ensure that its circumference never intersects the boundary of the flight corridor. If a non-adaptive approximation is used it would be possible for the error boundary to intersect with the flight corridor and the no fly zone.

For example, in-between the two London hospitals St Thomas and Guys, a flight corridor is defined by the area mainly above the Thames River, which is relatively safer to retrieve a fallen drone, see Figure 8. For a given flight path, see Figure 9, the reported locations of the drone may be inside or outside the corridor. Their distance to the boundary of the corridor is indicated by a circle. If one reports any position inside the circle, the verification results would be guaranteed to be the same as reporting the exact location of the drone.

The reason for not reporting the exact location here is to reduce the bandwidth and the delay for live streaming, which has been identified as a bottleneck to adopt tamper-proof blockchain technology in the evaluation of RQ1.

\[
P(L(t) \in Z) < \theta
\]

where \( \theta \) is a threshold close to zero to indicate the error tolerance level.

Taking into account the current location and the spatial continuity and proximity, the criteria can be rewritten as:

\[
P(L'(t) \in Z) < \theta \iff P(L(t + \tau) \in Z) < \theta
\]

where \( L(t) \) is the location of the drone reported previously, which may not be as accurate as physical location in reality \( L(t) \).

To test the verification correctness with required predictability in adaptation, we introduce multiple levels of noise in the reporting function. Given a trajectory of drone flight locations (i.e., a flight path) and a zone, by reducing the accuracy of the reported locations, one can observe how likely the verification fails to report any non-fly zone violation of drone flights.

In the following experiment, we have generated 100 flight paths using a software the simulators Dronology [12], Dragonfly [13], [14] and a specialised airspace editing tool ArduPilot.\(^8\) Here is a basic algorithm to check compliance of zone by the geolocations on the flight paths of a duration \( T \), which provides a a probabilistic answer to each flight path:

\[
P(L(t) \in Z) = |L(\tau t) \in Z|/|T/\tau|.
\]

\(^7\)Rounding is a simple form of a general class of locality-sensitive hash algorithms [11]

\(^8\)http://ardupilot.org
For a compliance threshold $\theta$, one obtains a boolean to tell whether the flight corridor is followed. For self-adaptive reporting of the compliance check results, the monitor takes into account the nearest distance to the boundary $d$ to adjust the interval $\tau$.

There are also several strategies to choose in self-reporting. E.g., the approximation function $L'$ (e.g., how many decimal digits one uses in reporting the locations). According to the theory of locality-sensitive hashes, approximation may not reduce the accuracy of reporting with respect to the verification results.

C. EVALUATION ON REAL-LIFE SCENARIOS

First, we chose a flight path and a flight corridor informed by TfL and NHS Blood, as mentioned in the scenario of Section I. The scenario requires the delivery of emergency medical assets including blood and organs between three hospitals, St. Thomas (T) Hospital near the Waterloo Station, St Bartholomew Hospital (B), and Guys Hospital (G). Straightline distance between each pair of the three hospitals is less than 5km, while the road travel time each way is typically 60 minutes due to traffic congestion.

In consultation with TfL drone experts, it is much safer to fly drones over the water because it may be less likely to hit someone on the ground. With this in mind, the flight corridor is defined to maximise the use of the waterfront on the Thames, while trying to take off and land the drones on helicopter pads or green parks near the sending and receiving ends of the deliveries. Figure 8 shows a flight corridor between T (lat $=51.49828$, lon $=-0.12007$) and G (lat $=51.50378$, lon $=-0.08734$), utilising the waterfront, which is defined by a blue polygon enclosed the zone on the map. The blue markers on the map also show a possible flight path from T to G.

However, some of the markers went out of the flight corridor, hence may cause safety and privacy damages.

By increasing the intervals, some way points will not be reported, which might decrease the accuracy of verification. For example, Figure 10 shows a reported flight path by 5 times bigger intervals: although all of points were inside the flight corridor, in fact the report was erratic because some of the skipped way points were in fact outside the corridor (cf. Figure 9).

By simulating random turbulence in 100 flight paths between T to G, Figures 11 and 12 shows the accumulated error rates $P(L(t) \in Z)$ of various configurations, which are characterised by two adaptable reporting parameters, $\tau$ and $d$.

The self-adaptive reporting solutions could adjust $\tau$, $d$, or both.

Let the probability of compliance $P(L(t) \in Z)$ with the minimal interval be an oracle, in our case, every $\tau = 15$ seconds a way point. For the 20 minutes journey there were 80 way-points collected for each of the 100 flight paths between T and G. When $\tau$ increases by a unit of 15 seconds, we show the absolute error rates of $|P(L'(t) \in Z) - P(L(t) \in Z)|$ on the average of all the 100 drone flights.
in Figure 11. From these data, the trend is clear that the absolute error rates increases when the intervals increases in reporting flight data.

Next, when \( d \) are fixed to a certain constant value \( d_0 \), i.e., reporting the truncated geolocation to a fixed precision no matter the drone is far away from the boundary of the flight corridor zone or not. In this case, one can see in Figure 12 that the error rate of \( |P(L'(t) \in Z) - P(L(t) \in Z)\) increases, almost in proportion to the accuracy thresholds (i.e. the horizontal axis shows the distance threshold by multiplying a unit of every 0.00005 degrees or 300m).

![Figure 11. Climbing error rates of \( P(L'(t) \in Z) \) while increasing the intervals between way-points in a drone flight.](image1)

The self-adaptive reporting algorithm reports a more precise location only when the distance of the drone is closer to the boundary, otherwise it reports a less precise location. As a result, the error rates is reduced to zero, as shown in Figure 12. In other words, any point selected from inside the circle can approximate the exact location (i.e., the centre of the circle) with respect to the verification queries.

Finally, putting the two self-adaptive parameters together by considering both the increased intervals and the reduced precisions in the algorithm, we estimate the bandwidth required for communicating fewer, less precise geolocations.

![Figure 12. Increasing error rates of \( P(L'(t) \in Z) \) while decreasing the precision of way-points in flight data reports; while self-adaptive distance threshold can reduce the error rates to zero.](image2)

Figure 13 shows the number of digits required to report the geolocations per way-point. We aim to use minimal digits to approximate way points, while maintaining the approximation to be within a given radius to the centre and avoiding crossing the zone boundary. The vertical axis shows the total number of digits required to store the approximate geolocations. The horizontal axis shows the distance for a non-adaptive algorithm to choose an approximate point within that distance to the exact way points.

The distance, i.e. the radius of the circles centred around the way points, is shown in Figure 7 by a unit of 0.00005 degree or 30m. The radius values range from zero for the finest approximation to \(39 \times 0.000005 \text{ degree} = 1200\text{m}\) for the coarsest approximation, and the total number of digits required for the decreasing precision would also decrease, ranging from the maximum 64940 digits to the minimum 44336 digits.

As shown in Figure 12, on the other hand, using the self-adaptive reporting algorithm on all the geolocations where the approximation radius is adjusted according to the minimal distance to the zone boundary, the total number of digits required would become 47804, which is a reduction from 64940 by 46% compared to the non-adaptive reporting algorithm of the same level of accuracy.

V. RELATED WORK

It is an active research area to elicit requirements and sketch architectures for secure vehicles [15], including the Internet of drones [16]. However, the problem of regulating drones have specific requirements which also calls for specific architectures.

A. DRONE SAFETY

Dronology is a metaphor to simulate the crowd of drones through a combination of cyber and physical domain properties, such as airspaces, drone behavioural data, etc. [12]. Apart from simulation-based approaches [13], policies-based [17] and autonomy [18] approaches have also been proposed to control drone safety [19]. The goal of these approaches is to identify critical safety hazards where drones may collide with each other (i.e., collision avoidance), whilst our goal is in forensic readiness, i.e., to track the drones and...
log their situations as tamper-proof and reliable evidence for forensic investigations [20].

For due diligence it is mandatory to be unbiased and avoid any conflict of interests amongst stakeholders who do not necessarily trust each other. In that sense, blockchains technology seems to be an obvious choice, given their decentralised and trust-less design.

### B. BLOCKCHAINS

With respect to the LiveBox requirements, software-based blockchain technology [5], [7], [8] already offers (1) scalable verification of live streamed flight status and (2) regulation of drone behaviour. In doing so, it can already address two challenges of physical FDRs: software has no weight and remote storage survives the destruction of drones, that is why Ethereum was considered for writing smart contracts to represent drone data [9].

However, they are not sufficient to address the forensic-readiness problem because both Bitcoin and Ethereum protocols cannot yet achieve the real-time verification requirement economically (as indicated by our pilot experiments). Furthermore, they need to establish a continuous communication channel, which cannot be assumed for drones operating in environments where communications are often intermittent. Therefore, we propose to enhance DLT, among many possible choices of solutions, that allows forensically sound caching of data until communications are re-established.

### C. PROACTIVE VERSUS SELF-ADAPTIVE FORENSICS

**Proactive forensics** or **forensic readiness** has been proposed to selectively log the evidence so long as the reasoning of forensic claims is not affected [21]. However, there is a gap between proactive and self-adaptive approaches to forensics. In proactive forensic readiness, we aim to collect the data as expected or predicted in order to establish or refute certain given hypotheses defined beforehand. On the other hand, in self-adaptive forensics, the amount of data we aim to collect would depend on the context of the running system. The case of the LiveBox self-adaptive reporting algorithm is such an example. In this sense, the forensic-ready adaptation is a special form of ‘cautious adaptation’ whereby the defiant component adapts to the global needs (e.g., forensic readiness) of the system of systems [22].

Compared to self-adaptive systems to deal with uncertainty using feedback loops [23], our self-adaptive reporting mechanism already knows how to deal with the adaptable parameters to the zone boundaries, i.e., a ‘set-point’ where the observed and the predicted contextual factors are compared at runtime to decide on which control actions to take in the next step. In this paper, we identified two of such adaptable parameters: the time interval and the distance to boundary. Interval is a controllable parameter by the self-adaptive system, while distance is a contextual factor parameter.

### VI. CONCLUSION AND FUTURE WORK

A “forensic ready” system is one that is able to, and does, collect the information necessary to support the investigation of an incident in which that system is subsequently involved. A flight data recorder (a blackbox, for example), provides such forensic readiness for an aircraft in case it is involved in an accident. Drones cannot usually carry heavy blackboxes, so alternatives are needed. As shown by our pilot studies and simulations, the LiveBox service for implementing the forensic-readiness requirements for drones is feasible. The service is self-adaptive because the forensic evidence to be collected is the result of a trade-off between the cost and the benefits of alternatives and the associated risks.

In this paper, we evaluated two research questions using a real-life scenario of emergency delivery of medical assets using drones through a flight corridor along the Thames River between two hospitals in London. When the flight data was live streamed, we showed that it cannot be tampered with when blockchain technology (such as a smart contract) is used to record the flight data in a distributed ledger. In contrast, traditional distributed database technology fails to achieve the tamper-proof property required. However, given that distributed ledger technology (especially public blockchains) is limited in managing large throughput of transactions for drone flight data, we investigated a self-adaptive reporting algorithm to selectively disclose geolocations at adaptable intervals and precision. Using our algorithm, the probability to infer whether a drone is inside or outside a zone does not change, while the total number of digits required by the approximated geolocation is greatly reduced. In a real-life case study, we showed such reduction to be up to 46%, while a zero error rate in terms of the verification results can still be achieved.

With these promising results, however, one must be aware that forensic-readiness requirements are only part of the overall solution; that is, there is still a need to provide full security life-cycle support for drone deployments, managing security threats as appropriate, managing privacy concerns, and investigating forensically incidents of misuse or accident. To achieve this in the future requires substantial collaboration by a community of stakeholders including not only researchers, but also law makers, law enforcement authorities, drone vendors, and pilots themselves.

### ACKNOWLEDGMENT

The author would like to thank David Bush, Anthony Rushton from National Air Traffic Services (NATS), and Gareth Sumner and John Laughlin from Transport for London (TfL) for insights and feedback on early draft of the article.

### REFERENCES


ANDREA ZISMAN received the B.Sc. and M.Sc. degrees in computer science in Brazil, and the Ph.D. degree in computer science from the Imperial College of Science Technology and Medicine, U.K. She was a Professor with the School of Informatics, City University of London. She was a Research Fellow with the University College London, U.K., and a Software System Consultant, Developer, and Analyst. She received Parnas Fellowship from the Irish Software Research Centre (Lero). She is currently a Professor of computing with The Open University, U.K. She is active in the research areas of software and service engineering where she has published extensively. Her current research interests include adaptation of software systems; traceability of software artifacts and food supply chain; secure software engineering; service engineering, incorporating discovery, adaptation, composition, reputation, and trust of services; cloud computing; and the IoT-based systems. She has been a Principal and Co-Investigator in various EPSRC, European, and industry funded research projects. She has given tutorials in many conferences. She has served in organizing and program committees for various conferences and workshops, and acted as a Reviewer and Guest Editor for many international journals. She is a Vice-Chair of the IFIP Working Group 2.9, and an Associate Editor of the Elsevier Journal of Software and Systems and the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING JOURNAL.

BASHAR NUSEIBEH is currently a Professor of computing with The Open University and a Professor of software engineering and a Chief Scientist with the Lero—The Irish Software Research Centre. He is also a Visiting Professor with University College London (UCL) and the National Institute of Informatics (NII), Tokyo, Japan. He is a Fellow of the British and Irish Computer Societies and the Institution of Engineering and Technology, and a member of the Academia Europaea. He received the ICSE Most Influential Paper Award, the Philip Leverhulme Prize, the Automated Software Engineering Fellowship, and the Royal Academy of Engineering Senior Research Fellowship. He received the IFIP Outstanding Service Award and the ACM SIGSOFT Distinguished Service Award. He was a recipient of the Royal Society-Wolfson Merit Award and two European Research Council (ERC) Awards, including the ERC Advanced Grant on adaptive security and privacy. He was the Chair of the Steering Committee of the International Conference on Software Engineering (ICSE). He serves as the Editor-in-Chief of ACM Transactions on Autonomous and Adaptive Systems and an Associate Editor of the IEEE SECURITY AND PRIVACY MAGAZINE.

* * *