# Why software engineers and developers should care about blockchain technology

Applications beyond digital currency

**Dr. Sam Siewert**

Email: siewerts@erau.edu
Social profile: http://faculty.erau.edu/Samuel.Siewert, https://www.colorado.edu/ecee/sam-siewert
Job title: Assistant Professor–Software Engineering Embry-Riddle Aeronautical University, Prescott; Assistant Professor Adjunct–Computer Engineering, University of Colorado Boulder

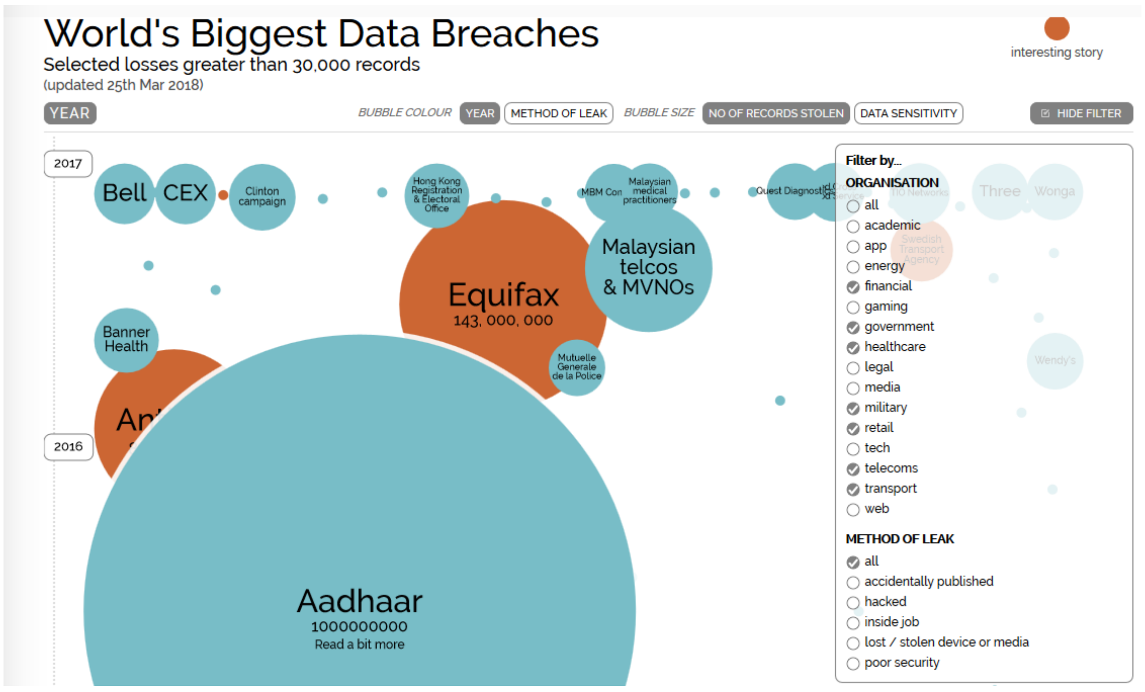
## Abstract

Blockchain is a revolution for data management that you can use instead of or in addition to a traditional structured relational database management system (RDBMS). Here, you dig into R-DBMS and blockchain (beyond use for crypto-currency and Bitcoin) by using a simple JavaScript blockchain prototype that you can expand to use Hyperledger to explore an emergent use for general aviation and small unmanned aerial systems tracking. IBM has helped to bring blockchain into much wider use for everything from supply chain logistics to medical information management (IBM blockchain). At Embry Riddle, I'm working on just such a database and comparing options including R-DBMS only, blockchain only, and a hybrid solution. I'm still working on determination of what will be best, but the more I learn about blockchain, the more interested I have become. While we have yet to settle upon a data management solution, we know we need to replace our flat files as our system grows, and we're most likely going to use a combined approach with both R-DBMS and blockchain distributed applications.

## A Brief History of Data Management and Processing

Relational database management systems (R-DBMSs) evolved as a formal method of manipulating data independent of data processing, the goal being to create a language (which became the Structured Query Language [SQL]) for querying (that is, to relate data in tables with primary and foreign keys, as shown in the Figure 4 schema for our working example) and to insert, update, and delete records in one or more tables consistently. IBM engineer E.F. Codd formalized relational algebra to reference data in related tables, which became core SQL. At the time, data integrity was a primary concern in addition to the ability to produce informational reports that combined data from normalized tables to support business decision making while avoiding loss of integrity caused by data hazards.

While this has made R-DBMS quite successful, at the same time, the centralized and concentrated data storage has made R-DBMS the focus of cyber-attacks, and lead to many data breaches as shown in Figure 1.



Filter by...

ORGANISATION
- ○ all
- ○ academic
- ○ app
- ○ energy
- ✓ financial
- ○ gaming
- ✓ government
- ✓ healthcare
- ○ legal
- ○ media
- ✓ military
- ✓ retail
- ○ tech
- ✓ telecoms
- ✓ transport
- ○ web

METHOD OF LEAK
- ✓ all
- ○ accidentally published
- ○ hacked
- ○ inside job
- ○ lost / stolen device or media
- ○ poor security

*Figure 1: Current recorded* RDBMS breaches*, 2016–2017*

So, why do R-DBMSs centralize all data in one or perhaps two spots (replicated only for a mirror or backup for disasters)? The goal is data integrity and information veracity. Data hazards are prevalent in files and spreadsheets, whereby related data becomes inconsistent through insert, modify, or delete operations. For example, say that an employee has been fired and deleted from human resources (HR) records but not from payroll. The RDBMS has been carefully designed so that with schema engineering, an experienced database engineer (DBE), and the process of normalization (breaking information up into tables of cohesive data, with primary key identification and relations through primary or foreign keys), the database will not become untrustworthy over time. Likewise, all transactions that query, update, or delete records meet atomicity, consistency, isolation, durability (ACID) criteria when SQL transactions are used with triggers (related updates) or stored programs. Centralization of data storage and access is fundamental to R-DBMS maintenance of referential integrity; likewise, one scalable server can provide data integrity and backup. That centralization, however, is also the main downfall of the RDBMS.

Centralized RDBMS server breaches in mission-critical enterprise systems have been growing, including in healthcare, transportation, telecommunications, government, and financial systems. Part of the problem is that all the sensitive data is concentrated in one place (perhaps replicated at a few other locations for disaster recovery mirroring). This organization is attractive to attackers and difficult to defend. It is inherently a single point of failure.

Now infamous examples of major centralized RDBMS breaches, such as Equifax, the US Office of Personnel Management, and Anthem, have motivated new approaches to data management and processing. The trend is growing and poses a major issue for RDBMSs, especially for transaction, log, and ledger data that is sensitive.

One approach is to patch and protect the current status quo, primarily with guards against network attack, SQL injections, and various tripwires and detectors based on artificial intelligence (AI) for unexpected activity as well as general training and protection for social hacking (phishing, spear-phishing, etc.). In addition, good RDBMS design practices help. A motivation for blockchain is to rethink R-DBMS based on well-known weaknesses (for example, impedance mismatch between client data processing and use and server data management and manipulation) and explore new applications that integrate data processing and data management or manipulation with a blockchain ledger approach.

This article explores the use of a Hyperledger blockchain (with a simpler JavaScript example) and a traditional R-DBMS for an application to log aircraft and drone locations. This is a real problem that the US Department of Transportation (USDOT) and the Federal Aviation Administration (FAA) face. The number of general-aviation (GA) aircraft in the United States is more than 200,000. Since the advent of Part-101 and Part-107 regulation allowing for hobby and commercial use of drones in class-G (uncontrolled) and with waivers in class-D airspace (restricted, such as near airports), FAA has also started to register small unmanned aerial systems (UASs). Ideally, FAA would like to allow for sharing of airspace between piloted GA aircraft and drones (small UASs).

## General Aviation and small UAS integrated data management

Recent FAA forecasts call for millions of aerial systems. FAA envisions that all compliant GA aircraft will make their current position and identification (International Civil Aviation Organization [ICAO] and tail number) known through an Automatic Dependent Surveillance–Broadcast (ADS-B) transmitter by 2020, perhaps along with Part-107 small UASs used for commercial operations such as package delivery or inspections systems. This change could mean millions of aircraft logging flight plans and current location regularly. Could this be a good application for blockchain?

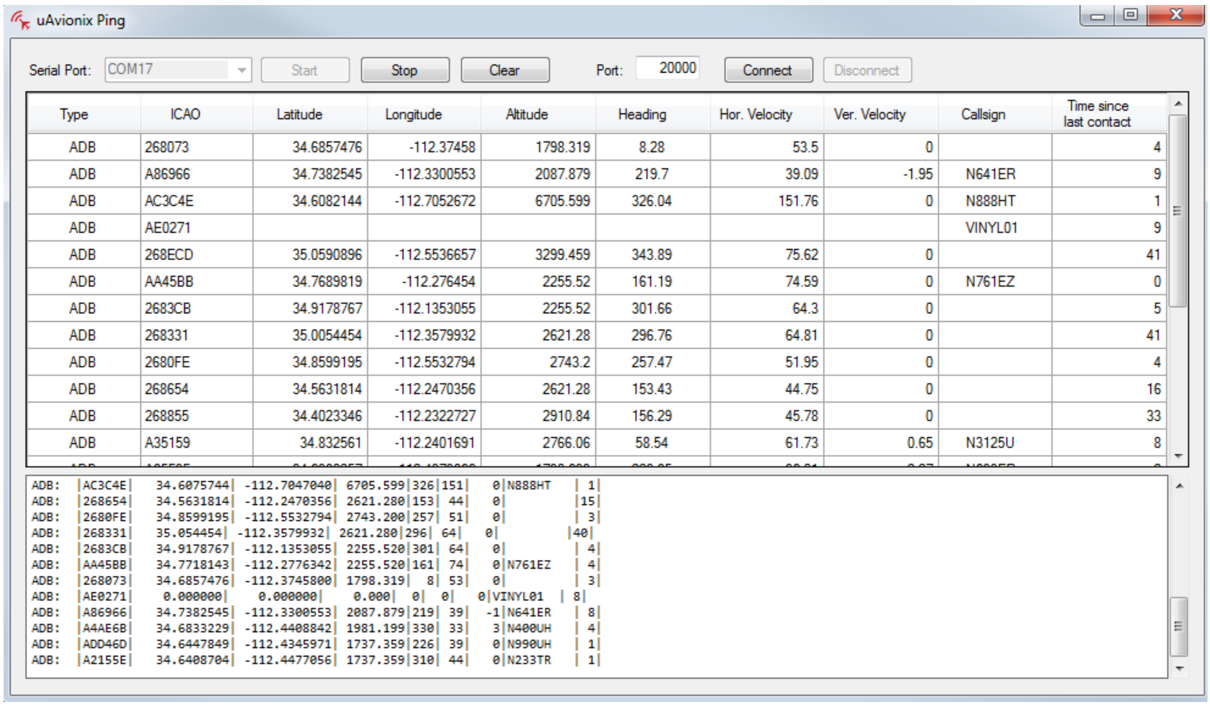
One key feature of this application is that it includes both traditionally centralized data from FAA registration of GA with N-numbers (tail numbers) cross-referenced with ICAO registration numbers and details of the aerial vehicle registrant (own or pilot), relational owner and pilot information, certifications, description of the air vehicle, etc., all of which is most often kept in an RDBMS. However, flight plans are filed as transactions, and ADS-B tracking from a flight in progress is logged by ADS-B receivers in a distributed network of ground stations and aggregated. So, although the basic concept of registering air vehicles, filing flight plans, and then tracking aircraft with global positioning system (GPS) and ICAO/tail number seems simple, it involves aggregation of sensitive data that could in fact be spoofed and provide to distributed users (other pilots in flight and ground stations).

Ideally, the information systems that GA and future Part-107 UAS operators use should be private, secure, consistent, protected against forgery (spoofing), and auditable in case of an accident or missing aircraft. From a cybersecurity viewpoint, a distributed system is not only natural for the system design (inherently distributed) but also can be more secure than concentrating all the data in one or a few RDBMSs and file servers.

## Tracking Compliant and Non-compliant Drones

Compliant small UAS (drones), especially commercial use, or Part 107 drones, are expected to provide identification and tracking via GPS (Global Positioning System) localization which they re-transmit to ground stations and other aircraft (and drones) via the ADS-B radio-frequency system as shown in Figure 2.  The main drawback of this approach is that the range is typically limited to 100 nautical miles.

*Figure 2: Example ADS-B Event Log from* Microavionix Receiver



| Type | ICAO | Latitude | Longitude | Altitude | Heading | Hor. Velocity | Ver. Velocity | Callsign | Time since last contact |
|---|---|---|---|---|---|---|---|---|---|
| ADB | 268073 | 34.6857476 | -112.37458 | 1798.319 | 8.28 | 53.5 | 0 | | 4 |
| ADB | A86966 | 34.7382545 | -112.3300553 | 2087.879 | 219.7 | 39.09 | -1.95 | N641ER | 9 |
| ADB | AC3C4E | 34.6082144 | -112.7052672 | 6705.599 | 326.04 | 151.76 | 0 | N888HT | 1 |
| ADB | AE0271 | | | | | | | VINYL01 | 9 |
| ADB | 268ECD | 35.0590896 | -112.5536657 | 3299.459 | 343.89 | 75.62 | 0 | | 41 |
| ADB | AA45BB | 34.7689819 | -112.276454 | 2255.52 | 161.19 | 74.59 | 0 | N761EZ | 0 |
| ADB | 2683CB | 34.9178767 | -112.1353055 | 2255.52 | 301.66 | 64.3 | 0 | | 5 |
| ADB | 268331 | 35.0054454 | -112.3579932 | 2621.28 | 296.76 | 64.81 | 0 | | 41 |
| ADB | 2680FE | 34.8599195 | -112.5532794 | 2743.2 | 257.47 | 51.95 | 0 | | 4 |
| ADB | 268654 | 34.5631814 | -112.2470356 | 2621.28 | 153.43 | 44.75 | 0 | | 16 |
| ADB | 268855 | 34.4023346 | -112.2322727 | 2910.84 | 156.29 | 45.78 | 0 | | 33 |
| ADB | A35159 | 34.832561 | -112.2401691 | 2766.06 | 58.54 | 61.73 | 0.65 | N3125U | 8 |

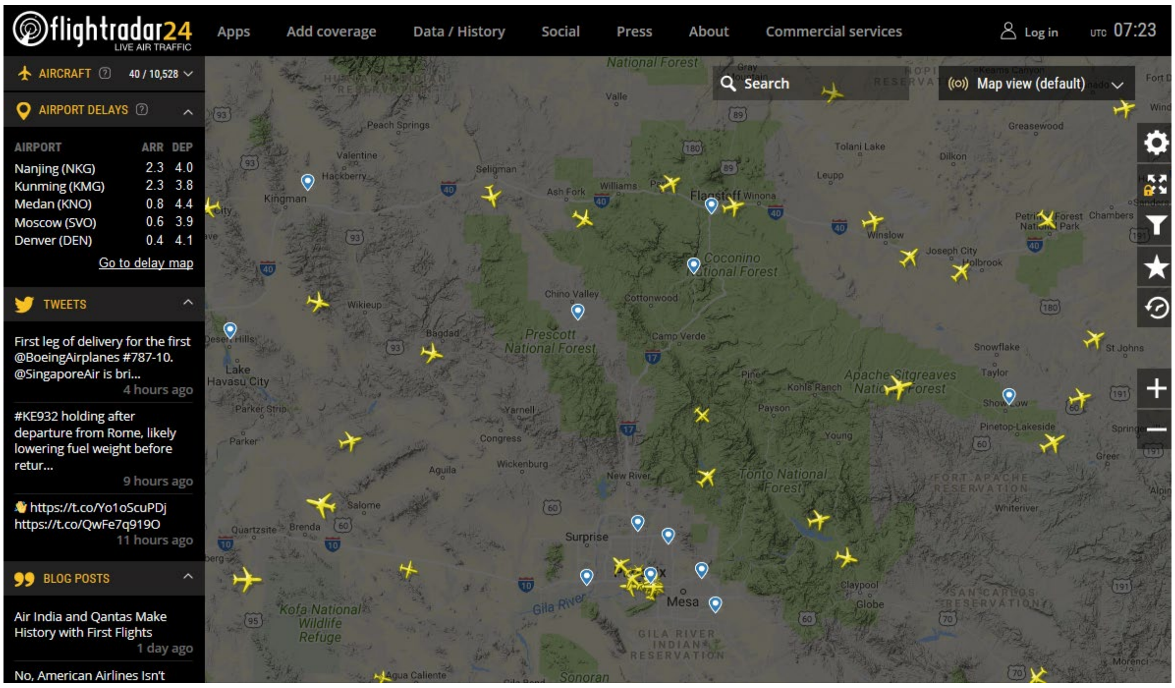
```
ADB:  |AC3C4E|   34.6075744| -112.7047040| 6705.599|326|151|    0|N888HT    | 1|
ADB:  |268654|   34.5631814| -112.2470356| 2621.280|153| 44|    0|          |15|
ADB:  |2680FE|   34.8599195| -112.5532794| 2743.200|257| 51|    0|          | 3|
ADB:  |268331|   35.054454| -112.3579932| 2621.280|296| 64|    0|          |40|
ADB:  |2683CB|   34.9178767| -112.1353055| 2255.520|301| 64|    0|          | 4|
ADB:  |AA45BB|   34.7718143| -112.2776342| 2255.520|161| 74|    0|N761EZ    | 4|
ADB:  |268073|   34.6857476| -112.3745800| 1798.319|  8| 53|    0|          | 3|
ADB:  |AE0271|    0.000000|    0.000000|    0.000|  0|  0|    0|VINYL01  | 8|
ADB:  |A86966|   34.7382545| -112.3300553| 2087.879|219| 39|   -1|N641ER    | 8|
ADB:  |A4AE6B|   34.6833229| -112.4408842| 1981.199|330| 33|    3|N400UH    | 4|
ADB:  |ADD46D|   34.6447849| -112.4345971| 1737.359|226| 39|    0|N990UH    | 1|
ADB:  |A2155E|   34.6408704| -112.4477056| 1737.359|310| 44|    0|N233TR    | 1|
```

The use of ADS-B by GA pilots is intended to reduce the need for ground RADAR and ATC (Air Traffic Control) and at the same time improve safety by allowing pilots to be observed by other pilots in a 100 nautical mile radius and to likewise have situational awareness themselves. Ground systems receivers can track air traffic and aggregate data to compare to filed flight plans. The basic transmitter works by receiving GPS (to locate the aircraft with the transmitter) and then broadcasting out location information (altitude, latitude, longitude) along with identification (ICAO and N-number) every few seconds. Logs of ADS-B would allow for NTSB investigators to know if an aircraft was off flight plan if it goes missing or crashes and enables much more intelligent management of air traffic in various classes of controlled and uncontrolled airspace.

With more than 200 thousand operating aircraft upgrading to ADS-B for safety and by requirement in 2020 along with potential small UAS use of ADS-B, this could generate quite a bit of data correlated to FAA GA and UAS registrations. In general, ADS-B is intended to support flight management to de-conflict flight plans that may change due to weather and to allow for UAS to give right of way to GA (based on FAA Part-107 regulations). Innovative use of drones includes package delivery, potential human autonomous air taxis, and a myriad or remote sensing (e.g. agriculture) and safety and security uses (e.g. search and rescue). While there have not been documented accidents involving drones and GA, aircraft have been grounded or waived off, sometimes in emergency scenarios such as firefighting due to potential conflict with illegal drone operations [Wildfire Today story].

The FAA has promoted registration [FAA registration], certification of remote pilots, and has created interim regulations and policies (part 101 and part 107) while NASA and the FAA jointly research how to best integrate small UAS with GA. Much of the UAS traffic management concepts for operations involves information and data management that must scale, be secure, private, but easily distributed and used by GA pilots, ground stations and UAS remote pilots for planning, in real-time and by NTSB and law enforcement for forensic investigation. The concept for integration of UAS with general aviation in the national airspace includes significant data management, processing, manipulation and distribution requirements for what NASA calls UTM or UAS Traffic Management [NASA UTM]. Companies and universities are working on UTM ground and flight technologies with NASA in the UPP (UTM Partner Program). Data collected from many ADS-B ground stations and in-flight receivers can be aggregated into databases such as the flightradar24 web page shown in Figure 3.

*Figure 3. – Example aircraft tracking map – Aggregated with ADS-B data sharing*



| AIRPORT | ARR | DEP |
|---|---|---|
| Nanjing (NKG) | 2.3 | 4.0 |
| Kunming (KMG) | 2.3 | 3.8 |
| Medan (KNO) | 0.8 | 4.4 |
| Moscow (SVO) | 0.6 | 3.9 |
| Denver (DEN) | 0.4 | 4.1 |

First leg of delivery for the first @BoeingAirplanes #787-10. @SingaporeAir is bri...
4 hours ago

#KE932 holding after departure from Rome, likely lowering fuel weight before retur...
9 hours ago

https://t.co/Yo1oScuPDj https://t.co/QwFe7q919O
11 hours ago

Air India and Qantas Make History with First Flights
1 day ago

No, American Airlines Isn't

The picture in Figure 3 is based upon the current air traffic control system and presently does not include small UAS. At Embry Riddle, we are working on a system to track both compliant and non-compliant small UAS traffic along with general aviation, with a network of ground stations to share and aggregate this data form ADS-B, active ground sensors (RADAR) and passive sensors (Electro-optical, Infrared, Acoustic and Radio Spectrum). This will require significant data sharing, data management and data processing. So, we have started to investigate our options.

## A comparison of R-DBMS and Blockchain

Before we decide on how to create our integrated aviation database system for identifying and tracking air vehicles, I want to review key differences between RDBMSs and blockchain. Both blockchain and RDBMSs provide data integrity and information veracity features, but the fundamental difference is that an RDBMSs is centralized, with a single or maybe a few servers; in contrast, blockchain is a highly distributed application with many servers. First, let's review R-DBMS.

### *RDBMS Characteristics*

The core characteristics of R-DBMS transactions, known as ACID, can wrap any SQL block of commands and include:

- **A = Atomicity.** Transactions are all or none, with rollback of all commands when they can't be completed.
- **C = Consistency.** A database is transitioned from one valid state to another (for example, employee who has been terminated is not left on payroll).
- **I = Isolation.** Concurrent execution of transactions results in the same state obtained if all transactions were executed sequentially.
- **D = Durability.** When a transaction has been committed, it will remain committed even if the system subsequently crashes.

The R-DBMS uses two methods to track data: (1) time-stamping and (2) a two-phase commit, which uses two-phase locking with rollback of transactions that can't be fully completed. This process is more fully described in the MySQL documentation for the InnoDB engine.

In an R-DBMS, data integrity is provided by both the physical and logical design. The physical design supports the logical and is most often based on redundant array of independent disks (RAID) and data mirroring or Exclusive OR (XOR) parity encoding. The logical design starts with record (tuple) attribute domains (type and range checking of attributes for inserted data); normalized table (schema) design and manipulation of data with transactions that meet ACID criteria described above. Any multi-table query and update relies upon both normalization and transaction wrappers to avoid inconsistent analysis, stale data, lost update and other well-known data hazards (documented well in R-DBMS textbooks such as Conolly & Begg).

Referential integrity is preserved based on (1) normalized table (schema) design, most often in the third normal form or a higher level of normalization or (2) primary key and foreign key relationships and triggers to update multiple tables (and stored programs).

Information veracity is supported by (1) SQL abstractions such as "views," or stored queries that provide reports in user-required formats without duplication of data in the database, or (2) by database administrator (DBA) monitoring of all database transactions and users (log combing).

## *Centralized R-DBMS Advantages*

R-DBMS has a clear and standardized design for data definition and manipulation which has allowed this technology to support the majority of data centers with wide adoption compared to alternatives (e.g. NoSQL, OODBMS, plain file systems). Abstraction such as "views" or stored queries can minimize knowledge of schema design and structure for users to simplify use and hide details and data users don't need to access. Stored programs and triggers can help maintain referential integrity and transactions with ACID characteristics maintain consistent state. Centralized R-DBMS storage allows for back-up, RAID redundancy and scaling (mirroring and striping), and maintenance and R-DBMS can be asynchronously mirrored for disaster recovery. Centralization allows transactions and access to be monitored and incomplete or errant transactions can be rolled-back and re-played if needed given ACID all-or-none characteristics. Finally, updates to R-DBMS software or the schema are handled easily in one central location and can be handled without downtime using mirrors.

## *Centralized R-DBMS Disadvantages*

While the clean break between data definition and data manipulation with data processing provided by R-DBMS and SQL has enabled a thriving industry, the separation of processing lead also to one of the biggest challenges with R-DBMS known as the impedance mismatch – where data processing and manipulation are separated, which requires use of an R-DBMS connector to process data or extensions to SQL for stored programs. As a result, client connections can be exploited – clients must send SQL requests to the centralized R-DBMS and therefore attackers can also send malicious SQL "injections" masquerading as a client. To solve this issue, client connections must include an encrypted channel for security to preserve data privacy and to prevent injections. Further, the clients must authenticate with the server hosting the database (e.g. for the secure transport tunnel) and again with the R-DBMS server itself. All data is concentrated in one place, so a data breach can result in loss of privacy for potentially millions of records, and breaches have occurred many times as shown in Figure 1. Finally, R-DBMS privileges must be administered for each database, but often features require global grants of privilege to R-DBMS users (e.g. schema manipulation, data integrity features and stored programs that update, delete, and insert records to maintain a consistent state in complex transactions). Due to complexity, a DBA may be tempted to grant more privileges than strictly necessary.

### *Blockchain Characteristics*

The core characteristics of blockchain guarantee no duplicate transactions (known as *double-spend*), provide secure time-stamping with authorization, auditability and accounting for all transactions. The blockchain has no expiration and the hash sequence provides integrity protection, prevents replication, provides forgery protection and ensures consistency.

### *Blockchain Disadvantages*

Data systems built using blockchain are most often referred to as "Dapps" or distributed applications, which might in fact be distributed servers, but with blockchain, the data is not fully centralized in one or two locations, but rather distributed over many peer locations. The Dapps have a transaction ledger grows over time as nothing is ever deleted or modified, rather all data is simply appended. Updates to Dapp software or protocols can be challenging given that all distributed peers on a network providing services must be updated. No central authority is available for identity verification (digital identification and certification is required from a trusted authority – centralized server), so complete decentralization of all data is not always feasible. Consensus protocols amongst peer Dapps are complex and not easily changed (trust verification for transactions) [Byzantine Generals problem described by Leslie Lamport et al]. Finally, dependencies on third party centralized services should not be built into DApps if avoidable, and therefore Dapps can't easily use storage-as-a-service or other Cloud services and must be self-supporting.

### *Blockchain Advantages*

While blockchain Dapps have some clear challenges, they can solve key problems that have really come to plague the modern R-DBMS suffering impedance mismatch (separation of data definition/manipulation and processing). Specifically, unlike the inherently centralized R-DBMS, the Dapp has no single point of failure and can prevent censorship or global blocking of service. Furthermore, the Dapp is not controlled by single authority (single point of failure) so trust can be higher (e.g. not subject to IT forgetting to remove a guest login or other security weakness from a single server that have caused epic R-DBMS breaches). Finally, the Dapp prevents double-spending (more generally duplicate transactions, e.g. spoofing and malicious data corruption is easily detected when a chain is broken).

## Traditional R-DBMS Schema Design and Data Insertion

The following is a simple SQL example for a log of ADS-B events – location, time, and identification for aircraft. The fully SQL implementation can be found on my GitHub along with a comparison block-chain prototype (https://github.com/siewertserau/drone-db). Figure 4 shows a MySQL Workbench schema for drone registration and tracking (simplified to show the most salient aspects). From an ER (Entity Relation) model such as this, it is possible to automatically generate SQL to instantiate the database schema. The example auto-generated code is included in the GitHub example code for comparison with programmer generated SQL.

Schema design and design tools have been a strong point of R-DBMS along with query support for reporting. The relational model minimized data duplication (limited to primary and foreign keys) and provides many protections against accidental data integrity and information veracity degradation over time.

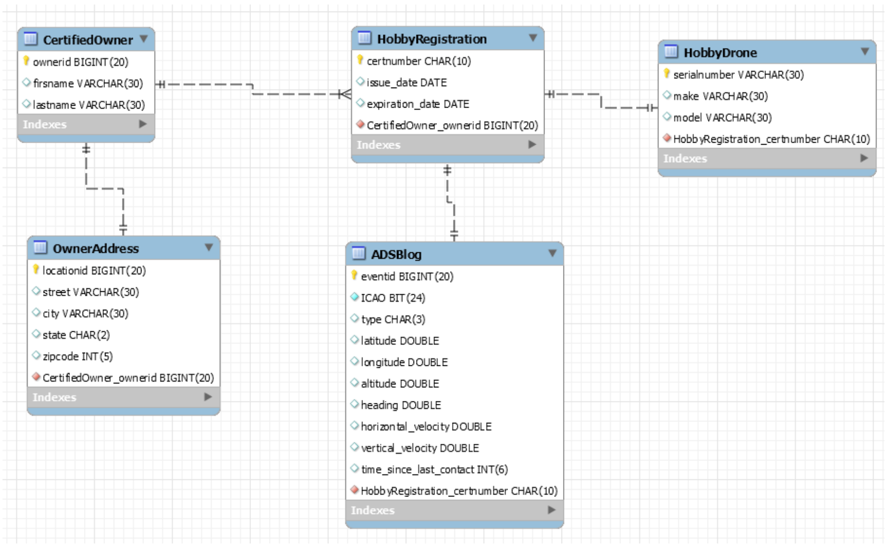


*Figure 4. – R-DBMS Schema for an ADS-B log (example to be updated once example comparison design is completed)*

The R-DBMS I'm working on has the following basic requirements including:

- Each aircraft has attributes with a primary key that is the ICAO or FAA registered tail number.
- The schema includes an aircraft owner table, aircraft manufacturer table, active tracking location information while operating (while transmitting ADS-B in flight).
- The database will allow for reports on compliant small UAS operation as well as small UAS tracked that were non-compliant (not registered, no flight plan, operating on controlled airspace without a waiver, etc.).

The SQL for schema to meet these requirements was developed into the following SQL DDL (Data Definition Language), which can also be automatically code generated. First the database and ADS-B log table are defined:

```
CREATE DATABASE dronetracking;
USE dronetracking;

CREATE TABLE ADSBlog (eventid BIGINT UNSIGNED NOT NULL AUTO_INCREMENT, ICAO
BIT(24) NOT NULL, type CHAR(3), latitude DOUBLE, longitude DOUBLE, altitude
DOUBLE, heading DOUBLE, horizontal_velocity DOUBLE, vertical_velocity DOUBLE,
callsign CHAR(6) NOT NULL REFERENCES HobbyRegistration (certnumber),
time_since_last_contact INT(6) UNSIGNED, PRIMARY KEY (eventid));
```

Then the registration tables (schema) are added to provide owner and small UAS identification for ADS-B tracking data logged as follows:

```
CREATE TABLE HobbyRegistration (certnumber CHAR(10) NOT NULL, issue_date DATE,
expiration_date DATE, owner BIGINT REFERENCES CertifiedOwner (ownerid), PRIMARY
KEY (certnumber));
CREATE TABLE HobbyDrone (serialnumber VARCHAR(30) NOT NULL, make VARCHAR(30),
model VARCHAR(30), certnumber CHAR(10) REFERENCES HobbyRegistration (certnumber),
PRIMARY KEY (serialnumber));
CREATE TABLE CertifiedOwner (ownerid BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
firsname VARCHAR(30), lastname VARCHAR(30), PRIMARY KEY (ownerid));
CREATE TABLE OwnerAddress (locationid BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
owner BIGINT NOT NULL REFERENCES CertifiedOwner (ownerid), street VARCHAR(30),
city VARCHAR(30), state CHAR(2), zipcode INT(5), PRIMARY KEY (locationid));
```

The SQL for data insertion of registered aircraft owners is simple and includes for example:

```
INSERT INTO CertifiedOwner VALUES(NULL, 'Sam', 'Siewert');
INSERT INTO CertifiedOwner VALUES(NULL, 'Rumple', 'Stiltskin');
INSERT INTO CertifiedOwner VALUES(NULL, 'Johnny', 'Flyboy');
INSERT INTO OwnerAddress VALUES(null, 3, '123 somestreet', 'somecityusa', 'AZ',
'86301');
INSERT INTO OwnerAddress VALUES(null, 1, '456 someotherstreet', 'anothercityusa',
'AZ', '86302');
INSERT INTO OwnerAddress VALUES(null, 2, '789 anystreet', 'somecityusa', 'AZ',
'86303');
```

The SQL for FAA hobby small UAS registration by owner, normalized to avoid update hazards makes use of the ICAO or tail-number (data is never duplicated) as follows:

```
INSERT INTO HobbyRegistration VALUES('FA39XMPHAA', '2017-11-04', '2020-11-04', 1);
INSERT INTO HobbyRegistration VALUES('FA3A4EEL74', '2017-11-06', '2020-11-06', 1);
INSERT INTO HobbyRegistration VALUES('ABCD123456', '2017-01-01', '2020-01-01', 2);
INSERT INTO HobbyRegistration VALUES('FEEDME1234', '2017-12-25', '2020-12-25', 3);
```

The SQL for FAA hobby small UAS registration by drone serial, make and model uses the primary key from registration to relate this to the drone make and attributes for it.

```
INSERT INTO HobbyDrone VALUES('serial123abc', 'DJI', 'Mavic', 'FA39XMPHAA');
INSERT INTO HobbyDrone VALUES('serial456def', 'DJI', 'Inspire', 'FA3A4EEL74');
INSERT INTO HobbyDrone VALUES('serial123xyz', 'DJI', 'Spark', 'ABCD123456');
INSERT INTO HobbyDrone VALUES('serial456zzz', 'DJI', 'Phantom', 'FEEDME1234');
```

Finally, tracking data SQL for tracking event insertion based on real-time ADS-B updates as shown in Figure 2 from an ADS-B receiver.

So, for example, the R-DBMS event insertions for a landing drone would look like the following:

```
INSERT INTO ADSBlog VALUES(null, X'012ABC', 'ADB', 34.620919, 112.452880, 5196.9,
0.0, 22.0, -1.0, 'FA39XMPHAA', 1);
…
INSERT INTO ADSBlog VALUES(null, X'012ABC', 'ADB', 34.620919, 112.452880, 5146.9,
0.0, 0.0, -1.0, 'FA39XMPHAA', 1);
```

Creating a block-chain for logging of ADS-B events that localize a specific drone based on a tracking station observation, it is imagined that the block-chain would be a sequence of tracking localizations.

To prototype this concept, I used JavaScript with WebStorm, JSON, yarn and NPM packages based upon tutorials readily available on YouTube. I adapted examples in my own JavaScript blockchain to capture data associated with ADS-B and to create a credit system for verifying tracking sequences. The JavaScript and corresponding SQL above can be downloaded from my GitHub (https://github.com/siewertserau/drone-db).

The mining would involve verification that the sequences are valid tracks of a registered drone or a non-compliant drone. Exactly how this might work and how block-chain provides added value requires further research, but some of the goals are:

- Tracking stations can share tracks of uniquely identified drones in a network.
- Tracking sequences can be shared between stations (and flight nodes) with assurance that tracks are valid (not faked).
- Data associated with tracking sequences can be traded using a credit system.

A tracking sequence, equivalent to R-DBMS insertions, could look like the following:

```
let adsbTrack = new Blockchain();

adsbTrack.createLocalization(new Localization('icao1', 'icao2', 1, 34.620919,
112.452880, 5196.9));
adsbTrack.createLocalization(new Localization('icao1', 'icao2', 1, 34.620919,
112.452880, 5191.9));
```

Assuming all ground stations and flight nodes transmit and receive ADS-B, the tracking localizations are a sequence that can be mined to create an overall tracking of a drone or drones by any node in a distributed network of tracking servers. When localization records from ADS-B (or perhaps from machine vision or RADAR) are created, they can then be mined into a verified block-chain sequence as follows for tracker-1:

```
console.log('\nStarting the tracker-1...');
adsbTrack.minePendingLocalizations('ground-station-address-1');
console.log('\nBalance of ground-station-1 is ',
adsbTrack.getBalanceOfAddress('ground-station-address-1'));
console.log('\nStarting the tracker-1 again ...');
adsbTrack.minePendingLocalizations('ground-station-address-1');
console.log('\nBalance of ground-station-1 is ',
adsbTrack.getBalanceOfAddress('ground-station-address-1'));
```

A trace of the block-chain JavaScript example will produce output including a full chain trace.  The first few lines of trace output are shown below and readers are encourage to run the JavaScript for a full trace example.

```
Starting the tracker-1...
Block mined: 0000afde409aeda76411cafc68e2c30734580257ea69bbab7bc81c94ad5d6169
Block successfully mined!
…
Current block: 0000a4d4ba2c0bd10f1650ae3c0dda4a602fd0de5024fb62037b25495b194bf8 ;
00006d9e633a9ce593e4219a9103e900d3a2e0fdaf152c38e04f786db0d9584e
Previous block: 00006d9e633a9ce593e4219a9103e900d3a2e0fdaf152c38e04f786db0d9584e ;
000003023631c65bf5c422f19b5f5ffba245685781e5685fb07b2db8081c9483

Is blockchain valid? True
```

The block-chain JavaScript would need significantly more work to be as fully functional as the SQL R-DBMS, but has advantages in terms of protection against corruption, built-in integrity checking with the hash-sequences and could be more completely implemented using more advanced block-chain frameworks like Hyperledger.

In general, my attraction to block-chain as an alternative to R-DBMS is to avoid having all drones, drone pilots, tracking stations, and many others directly accessing one centralized database via many connectors, which could lead to inadvertent disclosure of personal information of drone owners and operators. At the same time, the tracking and identification must be reliable and trustworthy and in fact provide evidence of legal operation of drones (and in some cases, illegal and forensic investigation for non-compliant use). Even if a centralized registration database can't be fully eliminated, the use of the block-chain could minimize central R-DBMS queries and thus minimize potential for a breach of the registration database.

## Conclusion

For specific applications that can benefit from the properties of blockchain Dapps, the use of blockchain has distinct advantages compared to centralized R-DBMS. Both can scale, but blockchain has inherent privacy, fault tolerance, double-report (double-spend) spoofing protection, and many more features that make it potentially superior to R-DBMS. Another option is a combined approach with a Dapp ledger and centralized authentication and records, which to a degree defeats the purist Dapp design by requiring a sing point of failure and nexus of control, but could provide a practical compromise.

Digital currencies are quickly adapting to trading and management on standard exchanges (https://bitcoin.org/en/exchanges) and are likewise making compromises in terms of combined Dapp and centralized accounts (wallet) and owner information management. The key in design of Dapp and hybrid information systems will be preserving the value of both approaches. A number of applications which require distributed networks of trust (e.g. the aircraft registration and tracking example in this paper, or for example credit checks and loans, healthcare logs, supply chains, and many other types of ledger and log oriented applications) can benefit from blockchain if carefully engineered. Some applications such as GA and UAS tracking and flight plan correspondence are inherently distributed by nature based on geography and distribution of assets.

Given the centralized R-DBMS data breaches in recent years, it appears that the blockchain new technology has arrived just in time to provide new options to secure data. Software engineers who want the advantages of Dapps combined with advantages of R-DBMS have a new opportunity to craft distributed servers for new and expanding applications with new technology to avoid data breaches and single points of failure.

## References

[1] E.F. Codd paper - https://cs.uwaterloo.ca/~david/cs848s14/codd-relational.pdf

[2] Byzantine Generals Problem – distributed trust and decision making - https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

[3] Hyperledger - https://www.hyperledger.org/

[4] MySQL – https://www.mysql.com/

[5] MySQL Workbench - https://www.mysql.com/products/workbench/

[6] FAA forecasts - https://www.faa.gov/data_research/aviation/aerospace_forecasts/

[7] FAA aircraft registration database - https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download/

[8] FAA UAS registration - https://www.faa.gov/uas/getting_started/registration/ , https://www.faa.gov/foia/electronic_reading_room/#geo_list

[9] Data breaches - http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

[10] Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.

[11] Flightradar24.com - https://www.flightradar24.com/

[12] Prusty, Narayan. *Building Blockchain Projects*. Packt Publishing Ltd, 2017.

[13] Swan, Melanie. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[14] Antonopoulos, Andreas M. *Mastering Bitcoin: Programming the Open Blockchain*. " O'Reilly Media, Inc.", 2017.

[15] IBM blockchain resources - https://www.ibm.com/blockchain/offerings.html

[16] ADS-B receivers and transmitters for General Aviation and Drones - https://www.uavionix.com/

[17] ICAO - https://www.icao.int/Pages/default.aspx

[18] ADS-B standard - https://www.faa.gov/nextgen/programs/adsb/

[19] Kopardekar, Parimal H. "Unmanned aerial system (UAS) traffic management (UTM): Enabling low-altitude airspace and UAS operations." (2014).

[20] UTM Partner program - https://www.nasa.gov/aeroresearch/programs/aosp/saso/partners

[21] https://www.jetbrains.com/webstorm/download

[22] https://nodejs.org/en/#download

[23] https://yarnpkg.com/en/docs/install

[24] Blockchain coding tutorials - https://www.youtube.com/watch?v=zVqczFZr124

[25] https://github.com/siewertserau/drone-db - example code in this paper

[26] WebStorm – https://www.jetbrains.com/webstorm/download

[27]JSON - http://json.org/

[28] NPM packages - https://nodejs.org/en/#download, https://www.jetbrains.com/help/webstorm/installing-and-removing-external-software-using-node-package-manager.html, https://yarnpkg.com/en/docs/install

[29] https://dev.mysql.com/doc/refman/5.5/en/innodb-transaction-model.html

[30] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).