Computers in Industry 126 (2021) 103393

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Building A blockchain-based decentralized digital asset management system for commercial aircraft leasing

Paul Kuhle[a,*], David Arroyo[b], Eric Schuster[c]

[a] Universidad Autónoma de Madrid, Facultad de Ciencias Económicas y Empresariales, Dept. of Finance, Calle Adam Smith 2, 28049 Madrid, Spain
[b] Instituto de Tecnologías Físicas y de la Información (ITEFI), Consejo Superior de Investigaciones Cientificas (CSIC), 28006 Madrid, Spain
[c] Technische Universität Berlin, Dept. of Aeronautics and Astronautics, Chair of Flight Guidance and Air Transport, Marchstr. 12, 10587 Berlin, Germany

ARTICLE INFO

Article history:
Received 25 June 2020
Received in revised form
10 November 2020
Accepted 29 December 2020
Available online 11 January 2021

Keywords:
Blockchain
Hyperledger
Aircraft leasing
Smart contracts
Privacy
Internet of Things

ABSTRACT

The blockchain technology has the potential to drastically change the way record-keeping and asset management is done. While more and more aerospace companies are starting to adopt the new technology, commercial aircraft leasing can still be considered uncharted territory for blockchain applications. The present paper analyzes how the sector stands to benefit from such a solution and proposes a suitable blockchain application design, focusing both on the regulatory and business needs as well as the implementation from a technological perspective. Our proof-of-concept demonstrates that asset management in commercial aircraft leasing can realistically be handled by a blockchain solution such as the one proposed in this paper.

1. Introduction

Commercial aircraft are often leased instead of purchased directly to either satisfy short-term demand spikes or as part of a long-term fleet planning strategy, where aircraft leasing is used as a tool to increase flexibility and free up capital (Kehoe and Hallahan, 2017). The importance of leasing contracts has been growing rapidly: By the end of the decade, roughly half of all commercial aircraft worldwide will be operated under a leasing agreement (IATA, 2017, p. 1).

In an environment where multiple parties operate with conflicting interests, trust cannot be taken for granted. It is therefore more than advisable to elude the dependency on only one or a reduced set of parties. Instead, the deployment of means to articulate distributed trust management should be promoted. Moreover, there is an urge, and even a normative and legal framework targeted to promote transparency and governance in commercial aircraft operations.

The blockchain technology has the potential to drastically improve the efficiency and security of supply chains in general (Forum W.E., 2019), and particularly in the case of commercial aircraft leasing. Our proposal for a decentralized solution builds upon trust-based approaches like the Global Aircraft Trading System (GATS) (Aviation Working Group, 2020). Specifically, we show how the blockchain technology can be leveraged to properly align the business model and the legal and normative requirements. This solution aims to mitigate the shortcomings and challenges of today's asset management solutions. These include inefficient processes, auditing issues and reduced asset management capabilities due to a lack of real-time information.

The objective of this paper is not to develop a software product or a blockchain application, but rather to propose a blueprint for a decentralized ecosystem for commercial aircraft leasing and to show how it integrates into the broader IT landscape in civil aviation.

In Section 2, we take a look at the current challenges that the industry is facing and how the blockchain technology can match the industry's business needs and regulatory requirements. We then present our proposal in Section 3.

2. Theoretical background

The following sections will give a brief overview of the key terms and ideas as well as the state of the art processes used in commercial

* Corresponding author.
E-mail addresses: paul.kuhle@inv.uam.es (P. Kuhle), david.arroyo@csic.es (D. Arroyo), eric.schuster@tu-berlin.de (E. Schuster).

https://doi.org/10.1016/j.compind.2020.103393
0166-3615/© 2021 Elsevier B.V. All rights reserved.

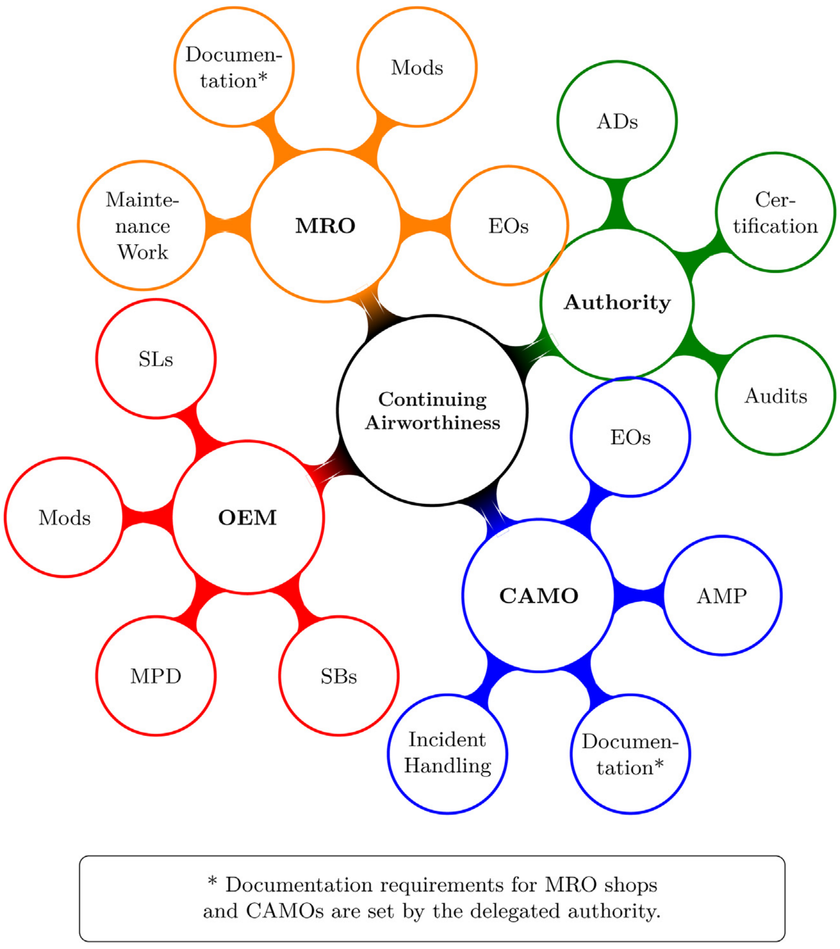


Fig. 1. Continuing airworthiness stakeholders.

aircraft leasing. First of all, it is discussed the regulatory environment (Section 2.1) and the individual stakeholders' responsibilities and needs in commercial aircraft leasing (Section 2.2). After that, the main challenges in commercial aircraft leasing are highlighted in Section 2.3 in order to be tackled by means of blockchain in Section 2.4.

2.1. *Commercial aircraft leasing and its regulatory environment*

Aircraft leasing only affects the operational phase of an aircraft's[1] lifecycle. In this phase, the operator's utilization of an aircraft is subject to *Continuing Airworthiness* (CA) regulations to ensure minimum operability. Legally, the operator is not allowed to assume responsibility for the technical condition of their own fleet. Thus, a multi-facetted stakeholder setup is created around the operator (see Fig. 1):

- the authority/regulator (and the delegated national authorities, e.g. the German *Luftfahrtbundesamt*),
- the Original Equipment Manufacturers (OEMs),
- the Continuing Airworthiness Management Organization (CAMO) set up or contracted by the operator of the aircraft and
- any Maintenance, Repair and Operations (MRO) shop that performs scheduled and unscheduled maintenance (MX) work on the aircraft, its components or its engines.

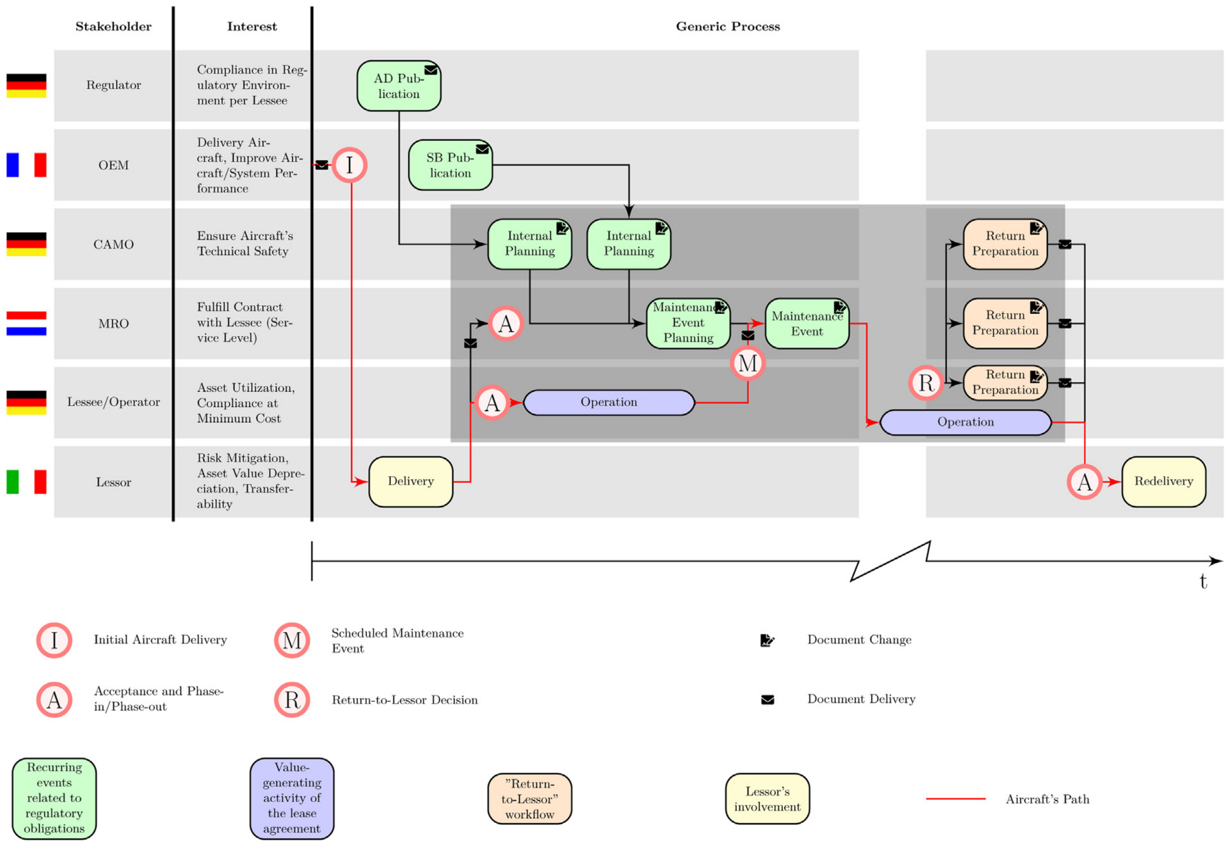
[1] A complex motor-powered aircraft as defined in (EC, 2008, Art. 3j).

In Europe, CAMOs and MRO shops are governed by European law and the European Aviation Safety Agency (EASA) (EU, 2014). The certification process is handled by the delegated authority which also oversees and audits all involved parties. Safety-critical findings are communicated to the authorities and the OEMs, who can issue the following publications:

- **Airworthiness Directives (ADs)** are published by authorities and contain mandatory instructions that need to be followed by the CAMO.
- **Service Bulletins (SBs)** are issued by the OEM and contain modifications and design improvements. SBs are not mandatory unless they are referenced in an AD. Other changes are often installed as modifications (Mods).
- **Service Letters (SLs) and Servive Information Letters (SILs)** are used to share non-critical information and documentation between the involved parties.

Consequently, different aircraft of the same type can vary significantly in equipment and component configuration. It is crucial to keep track of an aircraft's build stand, i.e. which ADs, SBs and other modifications have been installed. ADs may or may not affect an aircraft depending on its build stand, serial number or utilization.

Additionally, the CAMO is responsible for maintaining the aircraft in airworthy condition by following an Approved Maintenance Plan (AMP) on an organizational level and handling unscheduled incidents. AMPs can follow the Maintenance Planning Document (MPD) provided by the OEM. Any changes to the MPD must be approved by the authority, so that the CAMO's work complies with a



Stakeholder
Interest
Generic Process
Regulator
Compliance in Regulatory Environment per Lessee
OEM
Delivery Aircraft, Improve Aircraft/System Performance
CAMO
Ensure Aircraft's Technical Safety
MRO
Fulfill Contract with Lessee (Service Level)
Lessee/Operator
Asset Utilization, Compliance at Minimum Cost
Lessor
Risk Mitigation, Asset Value Depreciation, Transferability
AD Publication
SB Publication
Internal Planning
Internal Planning
Maintenance Event Planning
Maintenance Event
Operation
Delivery
Return Preparation
Return Preparation
Return Preparation
Operation
Redelivery
I
A
A
M
R
A
t
Initial Aircraft Delivery
Acceptance and Phase-in/Phase-out
Scheduled Maintenance Event
Return-to-Lessor Decision
Document Change
Document Delivery
Recurring events related to regulatory obligations
Value-generating activity of the lease agreement
"Return-to-Lessor" workflow
Lessor's involvement
Aircraft's Path

Fig. 2. Diverging interests of the stakeholders involved in commercial aircraft leasing across a simplified, generic process.

list of applicable MX processes, tasks and their intervals. Applicable tasks are carried out via Engineering Orders (EOs).

The MRO organization performs the actual MX work on the aircraft. The MX plan contains the scheduled Job Cards (JCs) and unscheduled Job Orders (JOs), and consists of line and base MX events, varying from 20 h to several months. MX events can cause diverse changes in aircraft composition depending on the amount and ratio of JCs and JOs solved per event (Transport Studies Group, 2008, Tables 1 and 2). After the MX work is completed, the aircraft is returned to the operator and put back into service.

The stakeholders can be based in different geographic locations and legal jurisdictions. This introduces more complexities with regards to non-standardized legislation, security concerns and privacy policies of the stakeholders' countries of origin. The exact requirements also depend on the lease type (dry or wet lease as defined in (EU, 2018, Art. 2, 24f.)). This paper focuses on a standard dry-lease agreement, as the stakeholder relationships of a wet-lease agreement are more trivial in comparison.

2.2. Business needs in commercial aircraft leasing and current state-of-the-art processes

The generic aircraft leasing cycle comprises a complex of stakeholders and processes which not always share interests and goals (see Fig. 2). After ordering an aircraft from an OEM, the lessor will take delivery of the aircraft and its documentation if all acceptance criteria are met. At this point, the lessor has complete knowledge of the technical configuration of the aircraft. Subsequently, the aircraft can be introduced into a lessee's fleet, at which point the asset leaves the lessor's legal jurisdiction.

Before aircraft (re)delivery, detailed inspections must be performed to guarantee compliance with financial, legal and with safety conditions (aircraft's physical condition, MX status and documentation).

During the lease, the lessee uses lessor-approved MRO shops to follow the AMP as agreed by both the regulator and the lessor. On the basis of the leasing contract, the lessors must document the aircraft's life to enable auditing of authority, responsibility and accountabilty. Internally, the lessee, CAMO and MRO organizations often use commercial MRO software like AMOS to share information about business processes like logistics, operational planning and controlling. This workflow (highlighted in black in Fig. 2) is opaque from the regulator's, lessor's and OEM's point of view. To gain insight into this black box, they can perform audits whose efficacy and efficiency depend on the organization's capability to produce the required records. Although extensions like FlyDocs endow MRO software to fulfill this goal, current implementations mainly rely on the digital replication of paper-based documentation.

A typical leasing contract goes beyond Fig. 2 and covers payments, risk mitigation and operational restrictions. These elements are usually negotiated freely between the lessor and the lessee and often involve additional stakeholders like insurance brokers, consulting firms, payment processors and other financial institutions. From an asset and lifecycle management point of view (Hanley, 2011, pp. 32–37), a leasing contract consists of four key elements: *Delivery and Redelivery*, *Operations*, *Payments* and *Risk* (see Fig. 3).

The lessee's monthly lease rate consists of a base rate and an additional rent. The additional rent covers the lessee's contributions to future MX events derived from the lessee's utilization of the aircraft and to be carried out after the lease expires (e.g. structural work is often carried out in intervals of 10 years or more, which greatly exceeds the average lease term).

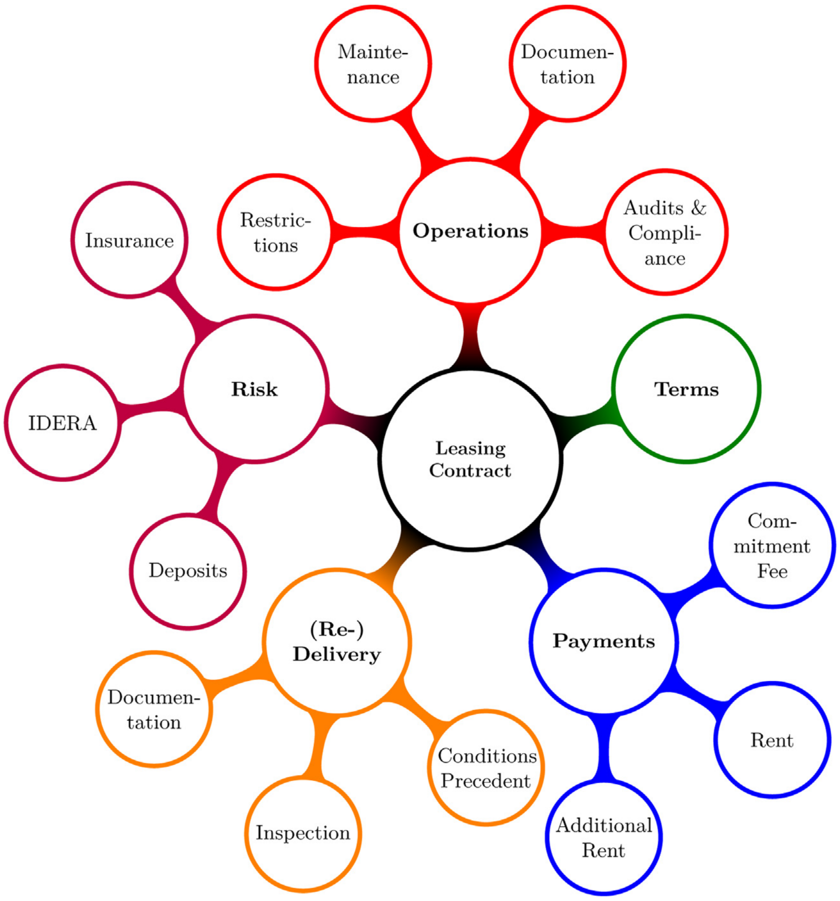


Fig. 3. Leasing contract components.

The three main tools to mitigate risk in a lease agreement are security deposits, insurance and the Irrevocable De-Registration and Export Request Authorization (IDERA), which provides an international legal regime to govern security interests in aircraft (ICAO, 2001). It allows creditors to repossess an aircraft in case the lessee defaults on his payments. Additional operational restrictions may be imposed by the lessor during the lease (e.g. the lessee is not allowed to change the habitual base of the aircraft to another country or to fly the aircraft into active warzones or states sanctionted by the United Nations Security Council).

While there have been efforts to standardize at least parts of aircraft lease agreements, most lessors have their own proprietary leasing contract templates (IATA, 2012). This paper will focus on a generic leasing contract and a generic timeline based on publicly available information.

2.3. Current challenges

The current approach to document-sharing carries a certain risk of fraud, document loss and auditing issues, even in developed countries (FAA, 2019). Signatures and stamps on physical documents and their digital replicas, if they exist, add little value to an external auditor. Company-internal identity management systems are not linked to government-issued identifications or external certifications that the auditor may require. Tracing the internal chain of responsibility in the organization seems impossible under today's bottom-up MX approach as company-internal processes cannot be monitored by outside stakeholders (i.e. regulators and lessors). Indeed, the current system is missing real-time information about the aircraft's operational and CA activity, which hampers the regulator's ability to ensure compliance with the MX obligations outlined in Section 2.1. As of today, the regulator has to rely on costly and, given the mentioned issues, potentially unreliable audits. This problem also affects downstream stakeholders like MRO organizations and suppliers that are subject to similar audits.

Besides operational safety, as part of the asset management strategy, the lessor must deliver the aircraft to the next lessee in airworthy condition. Due to the current lack of automation, the lessor first gets a chance to verify maintenance documents at lease expiry. Consequently, at this stage the exhaustive inspection of the aircraft and its documentation leads to a significant workload. According to the timeline proposed by IATA (IATA, 2017, pp. 58–61), the whole redelivery process takes more than a year for a long-term lease. Operational restrictions and maintenance reserves can often only be checked, enforced and collected when the aircraft is physically redelivered, even with IDERA.

Another concern arises from the selection of adequate Information Technology (IT) architectures for the distributed and sensitive nature of the aircraft leasing ecosystem. Current approaches like the GATS (Aviation Working Group, 2020) adopt a centralized management scheme, which could erode stakeholders trust in the IT system. Indeed, the stakeholders of our ecosystem often need to gather information from other stakeholders whose interests can be diametrically opposed.

A centralized global solution would likely have insufficient international backing. In fact, countries like Russia and Iran have been subject to sanctions affecting their civil aviation industry (Nadimi, 2019) and their trust in centralized global institutions like the United Nations and their suborganizations like the International Civil Aviation Organization (ICAO) is limited (Radin and

Reach, 2017, pp. 39). This solution could therefore only capture a portion of the market and will always be limited by the trust that other stakeholders (and even individual employees as part of a criminal liability in MX fraud or irregularities) have in the solution's operator.

2.4. Technological capabilities of blockchains in the context of commercial aircraft leasing

The functionality of the blockchain technology can help to tackle operational and tactical aspects in aircraft leasing, and also to bolster new strategical schemes. Blockchains are append-only lists of blocks in which validated transactions are stored chronologically (Narayanan and Clark, 2017). Each new block contains a certain number of transactions. When a block reaches that target number of transactions, it is added to that list, thereby generating a ledger of all conducted transactions. This ledger is not stored in a central database; instead it is stored on every peer in the network. Moreover, the information written in a blockchain cannot be easily unilaterally modified, which means that the information persisted in a blockchain can be regarded as immutable (unless enough peers collude).

Beyond the chain supply management in the airport industry (Di Vaio and Varriale, 2020), asset traceability and data provenance are fundamental in the management of aircraft leasing. The tamper-proof nature of the blockchain and the chronological recording in the blocks are very useful to foster audit and accountability procedures (Forum W.E., 2019). The capability of creating a trust layer that is independent from the rest of the communication protocols stack is of major relevance, since it enables disintermediation and reduces the dependency on third parties along the lifecycle of the leasing process. As it has been widely developed in financial networks (Yermack, 2017), blockchains endorse collaborative decision making and taking processes, which reduces the dependency on single entities or organizations. This is also of high relevance in the field of cybersecurity, where collaboration between actors must be conducted according to convenient data minimization schemes (Rubio et al., 2020).

Many airlines are already using blockchains, but only for specific tasks. For example, Russian-based S7 Airlines is handling more than \$1 million in monthly B2B payments, e.g. for aircraft refueling, via a blockchain platform (S7 Airlines, 2019). Blockchains can help to automate other business processes by means of smart contracts (IATA, 2018). Smart contracts are coded agreements that are stored in the immutable blockchain to be automatically executed once a set of conditions is satisfied. This capability enables machine-to-machine interaction, which is a requirement in the deployment of advanced solutions for the Internet of Things (IoT) (Lao et al., 2020). As another application context in the Industry 4.0, the aircraft leasing industry can benefit from the tokenization of assets and their management by smart contracts.

Decentralized approaches that encourage the efficient and safe sharing of information and documentation have the potential to drastically improve lifecycle and asset management in commercial aviation because they can provide trust between parties that otherwise have little reason to trust each other. Identity management is crucial to attain these goals, and it demands the articulation of robust and efficient access and authorization models. In this regard, we have to take into account that blockchains can either be public, private or a consortium (see Table 2). A *public blockchain* allows anyone, barring hardware restrictions, to set up a node on his computer and read data from the blockchain. *Private chains* use whitelist filters to make the data available only to selected individuals. In a private blockchain, these peers are typically controlled by one single company, making this type of blockchain less suited for truly decentralized applications. Some application contexts call for an adequate balance between decentralization and accountability. In this regard, a *consortium blockchain* (also known as hybrid blockchain) is a viable solution. Consortium blockchains are built around the idea of allowing access only to a few peers from different companies.

Finally, we have to bear in mind that blockchain is a means to persist information and to ease its sharing after being stored. The source of the information is external and, consequently, the interface using during the recording stage is not protected by the tamper-proof characteristics of the blockchain. Data curation in general (Heiss et al., 2019), and particularly in the case of legal compliance and liability (Arribas et al., 2020) demands the inclusion of off-chain actors and/or agents. Therefore, the reliability and trustworthiness of these actors should be further analyzed (Yang et al., 2020).

3. The proposal

The requirements for our concept are derived from the regulatory environment, the business needs and the challenges that the aircraft leasing industry is currently facing.

The functional requirements can be seen in Table 1. The non-functional requirements fall into the categories *Compliance* (Table 3), *Performance* (Table 4), *Business Needs* (Table 5) and *Usability* (Table 6).

In this section, we propose a blockchain architecture that implements these requirements.

3.1. Blockchain platform

This project calls for a consortium blockchain, given that commercial aircraft leasing is a multi-organizational application. The blockchain frameworks that could potentially be used to develop this type of application are Ethereum and Hyperledger Fabric or Sawtooth. For this application, Hyperledger Sawtooth can be recommended, as it can fulfill the non-functional requirements better than Hyperledger Fabric and Ethereum:

- Hyperledger is backed and being contributed to by industry leaders like Airbus, Intel, IBM and Consensys (Hyperledger, 2020), making Hyperledger frameworks more maintainable in the long term (**E-4**). The involvement of smaller IT companies and consulting firms like Accenture means that Hyperledger has already developed an enterprise ecosystem (**E-3**).
- A blockchain built with Hyperledger Sawtooth can incorporate customized nodes that can be written in a number of different programming languages. Sawtooth does not require a specific deployment setup or data access architecture. This reduces vendor lock-in and increases overall flexibility (**E-2**).
- The Proof of Elapsed Time (PoET) consensus protocol available in Sawtooth makes use of a much more efficient algorithm than Proof of Work, which is used in Ethereum. Since the algorithm is a major driving force behind the blockchain's overall power consumption (Li et al., 2019), the PoET protocol offers an advantage with regards to **D-2**.
- The PoET protocol is Byzantine fault tolerant (BFT), i.e. it can reach consensus even when the system has malicious or faulty nodes. This is not offered by Hyperledger Fabric's Kafka protocol. PoET is also more scalable than Kafka (The Hyperledger Project, 2017) (**C-1**).
- Sawtooth has built-in access control and user management, but allows the use of neutral data sources and off-chain storage (**D-1**).
- Given Hyperledger Sawtooth's compatibility with Ethereum smart contracts, crypto-payments and other public blockchain services can be integrated easily (**E-1**).

Table 1
Functional requirements.

Nr	Title	Description
A-1	Tokenization	The physical asset (the aircraft) will be *tokenized*, i.e. issued a digital counterpart.
A-2	User management	A user management system needs to be implemented to handle access rights and stakeholder interactions.
A-3	Process automation	The processes and interactions shown in Figs. 2 and 3 should be implemented as *smart contracts*.
A-4	Interfaces	The system needs to provide real-time access to relevant data. Regulators should be given access to MX records, while lessors should also be able to audit flight logs, equipment lists, the build stand, etc. Interfaces should be provided to the respective stakeholders.

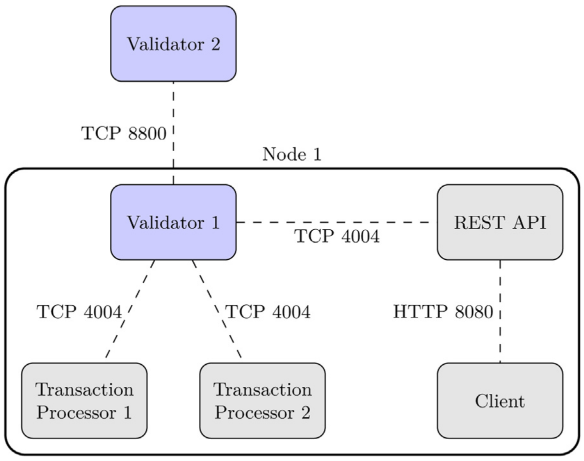


Fig. 4. Overview of Hyperledger Sawtooth Components. Adapted from (Intel Corporation, 2019).

Fig. 4 shows a generic overview of the Hyperledger Sawtooth components. Each node in the network contains a validator, one or more transaction processors (TP) and a REST API.

The validator contains the blockchain data and communicates with the other validators in the network to approve the transactions sent by the TPs. The TPs contain the business logic of the blockchain, i.e. functions to interact with blockchain data as well as the appropriate data models to encode and decode data that is stored on the blockchain.

In order to communicate with external applications, a Hyperledger Sawtooth node needs to have a REST API. The REST API handles transaction requests and event broadcasting, i.e. it sends out a signal to inform subscribers about state changes and other important events. This architecture makes it possible to tailor clients to the customer's needs.

3.2. Proposed design

3.2.1. Architecture

The architecture of the proposed solution can be seen in Fig. 5. Hyperledger Sawtooth already implements the protocol layer, including the decentralized data storage layer, the consensus mechanism and user management system.

The application layer contains the business logic as described in Section 3.1. This TP contains data models for the aircraft, the leasing contract and the AMP (**A-1**). Stakeholders can interact with these data models according to their individual interests and responsibilities outlined in Section 2 using their respective transaction families (**A-3**) as shown in Fig. 6. Transaction families may need to query external data sources as presented in Fig. 5. Custom APIs and clients are added as an abstraction layer (**A-4**).

3.2.2. User management

One of the key features is user management (**A-2**), which is commonly implemented as *Authentication, Authorization and Accountability (AAA)*. This application uses an asymmetric cryptography system to ensure robust user authentication, creating a public and private key pair for every stakeholder that has to interact with the blockchain. Once the user is authenticated, they may need authorization to perform certain tasks. Hyperledger Sawtooth provides three different methods to handle these permissions:

- **Allowed transaction families.** The network operator can specify which types of transactions can be submitted, making the network ignore unrecognized TPs. This application only recognizes the internal Hyperledger Sawtooth transaction families and the custom TP (*aero*) that was developed in this work.
- **Validator permissions.** This functionality can be used to specify the nodes that are able to connect to the network.
- **Transactor permissions**, which are used to define who can submit transactions to a validator. Permissions can be set off-chain (locally) or on-chain (network-wide). A transaction is only allowed if both the off-chain and on-chain settings agree that it should be accepted. For instance, an MRO shop may be given restricted transaction rights in the on-chain settings. The MRO shop could use off-chain settings to further restrict transaction rights within their own organization, e.g. to give engineers and mechanics different permissions.

For the sake of network security, stakeholders were only given transaction signer rights for the *aero* transaction family. Moreover, Hyperledger Sawtooth fosters accountability. First, the underlying blockchain stores all transaction receipts in an immutable and transparent way. Second, any single transaction is digitally signed and therefore it is possible to trace back actions to individual identities.

3.2.3. Data models

While all data is stored on-chain within the validator, data models are needed to make sense of it. The validator only stores serialized data, i.e. in a series of bits that is suitable for transmission and storage. Google's Protocol Buffer format is used as the serialization format. Protocol Buffers are language-neutral, platform-neutral, fast and lightweight (Google LLC, 2019). They also offer deterministic serialization, which is important given that the validator enforces deterministic behavior in the TP and data models.

In this application, the public keys used for user authentication are also used for address handling. Hyperledger Sawtooth uses 35-bit addresses to identify the position of each portion of data within the blockchain's data storage system (see Fig. 7). For most stakeholders, their public key can be used to generate a unique address. Other entities like aircraft do not have a public key, as they do not represent a stakeholder. These addresses need to be generated

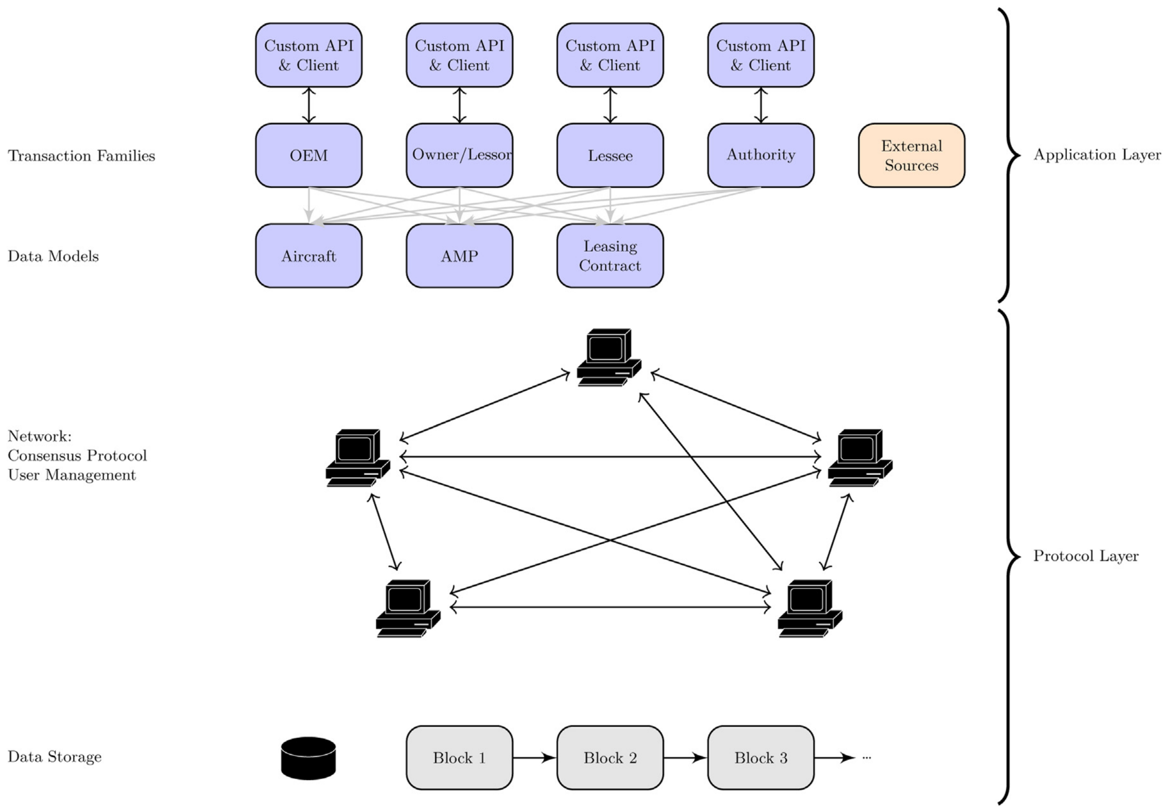


Fig. 5. Architecture overview.

using model data, e.g. by hashing the aircraft type and Manufacturer Serial Number (MSN).

The stakeholders have similar data models that store their public key, name and country. These models can be inherited from and expanded as needed to store additional information. As an example, the data model for an aircraft owner/lessor can be seen in Code Block 1.1.

The aircraft data model stores the stakeholder history (OEMs, owners, operators, authorities), the flight, equipment, MRO event, AD, SB, location, hours, cycles and audit histories as well as some basic data like its own address, MSN and engine type. It also stores a list of associated AMPs and leasing contracts. The model can be expanded as well, but the provided generic implementation in Code Block 1.2 is sufficient to fulfill **B-3**.

The AMP data model stores MX tasks and their applicability, thresholds and associated maintenance events. The implementation shown in Code Block 1.3 is fairly generic and can be used with any AMP.

As discussed in Section 2.2, there is no generic or industry-standard leasing contract. It would not be sensible to force a standardized contract scheme onto the lessor and lessee, who are free to negotiate their own terms, restrictions and conditions. Since the business relationship between the lessor and the lessee does not concern any other stakeholders, they need to be able to set up their own transaction families and data models. Sample contracts like the one used in this proof-of-concept (Code Block 1.4) or individual modules like geographic restrictions can be provided.

3.2.4. *Data storage*

Because blockchains are not meant to store large amounts of data, the data models are not used to store large files like PDF documents that are often necessary to properly document contracts, MX events and audits. Instead, the blockchain stores a link (or anchor) to the file. This anchor is usually contructed using a hash function (e.g. SHA-512), and it can be used to ensure that the file has not been modified after the initial upload.

For simplicity's sake, this paper assumes that all files are stored off-chain in a location chosen by the data owner (e.g. their own servers or a cloud solution like AWS). This location would need to be compatible with the Simple Storage Service (S3) protocol. A dedicated transaction family is then used to generate pre-signed URLs to protected S3 files. The key used to generate this URL would only be stored on the data owner's node. The blockchain would simply act as a consensus mechanism to determine access rights. This process is shown in Fig. 8.

This solution satisfies **B-1**, **B-2** and **D-1**. However, it is not a truly decentralized solution. While the blockchain determines access rights, the data owner could always overwrite this in his node's settings or in his storage server settings.

A better approach would be to employ a distributed file storage solution (Huang et al., 2020).

3.2.5. *Transaction processor*

As shown in Fig. 9, the TP consists of a transaction handler and a state handler. The transaction handler is responsible for deserializing the incoming transaction payload, verifying the received data and generating the container data that will be recorded on the blockchain. State changes are handled by the state handler, which serializes the container data according to the specifications in the data models and submits the state change.

The most complex and computationally expensive parts of the process are the verification of transaction data and the generation of new container data. For instance, when a new flight is added to the aircraft's flight log, the transaction handler has to verify the items shown in Fig. 10.

Some of these items require further input from neutral data sources (orange boxes) or other blockchain data models (green boxes). The transaction handler can invoke other TP functions, e.g.

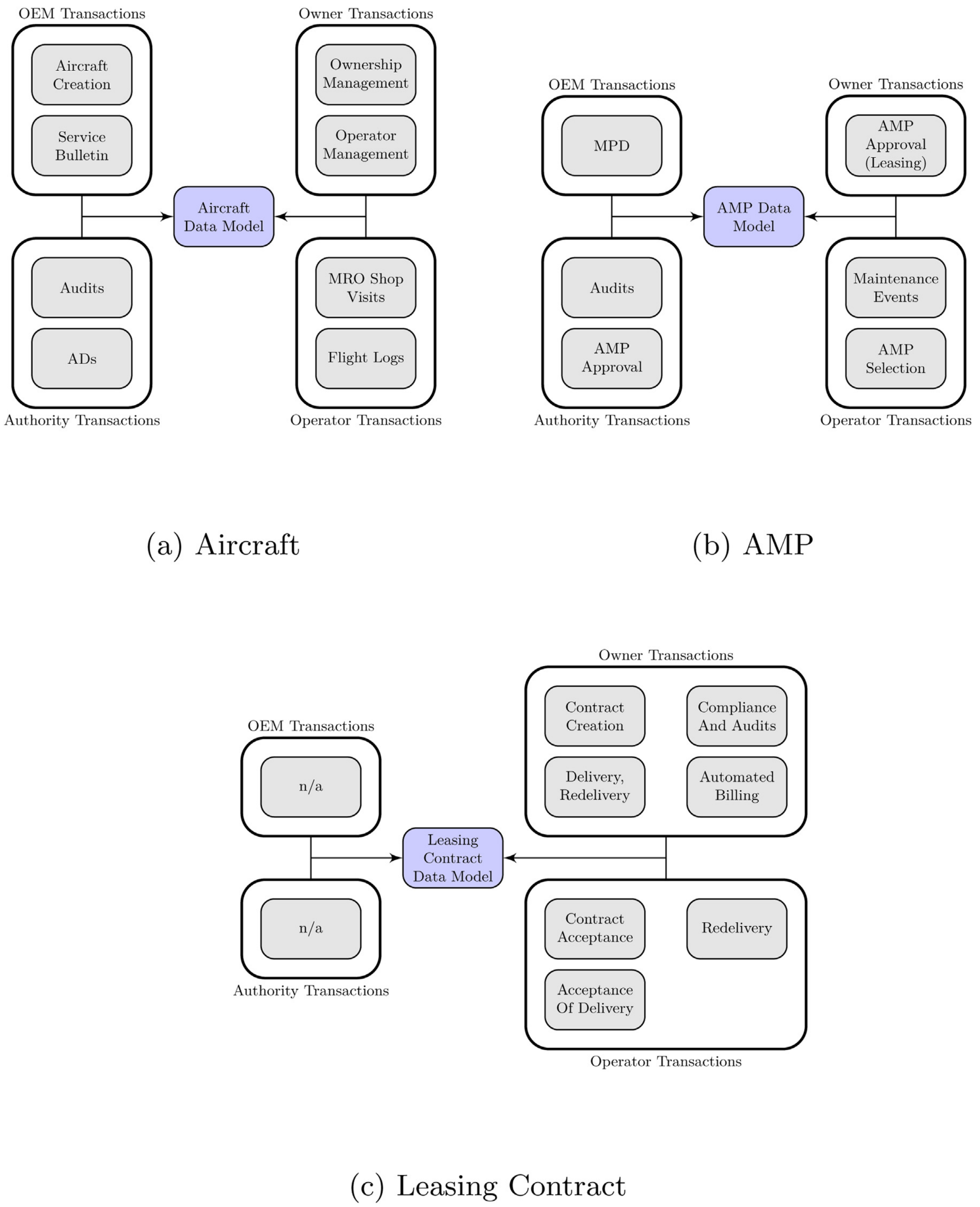


Fig. 6. Data model interactions.

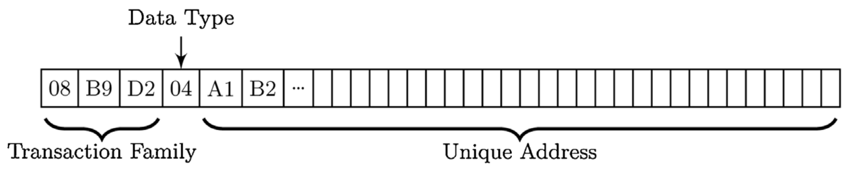


Fig. 7. Address handling in Hyperledger Sawtooth.

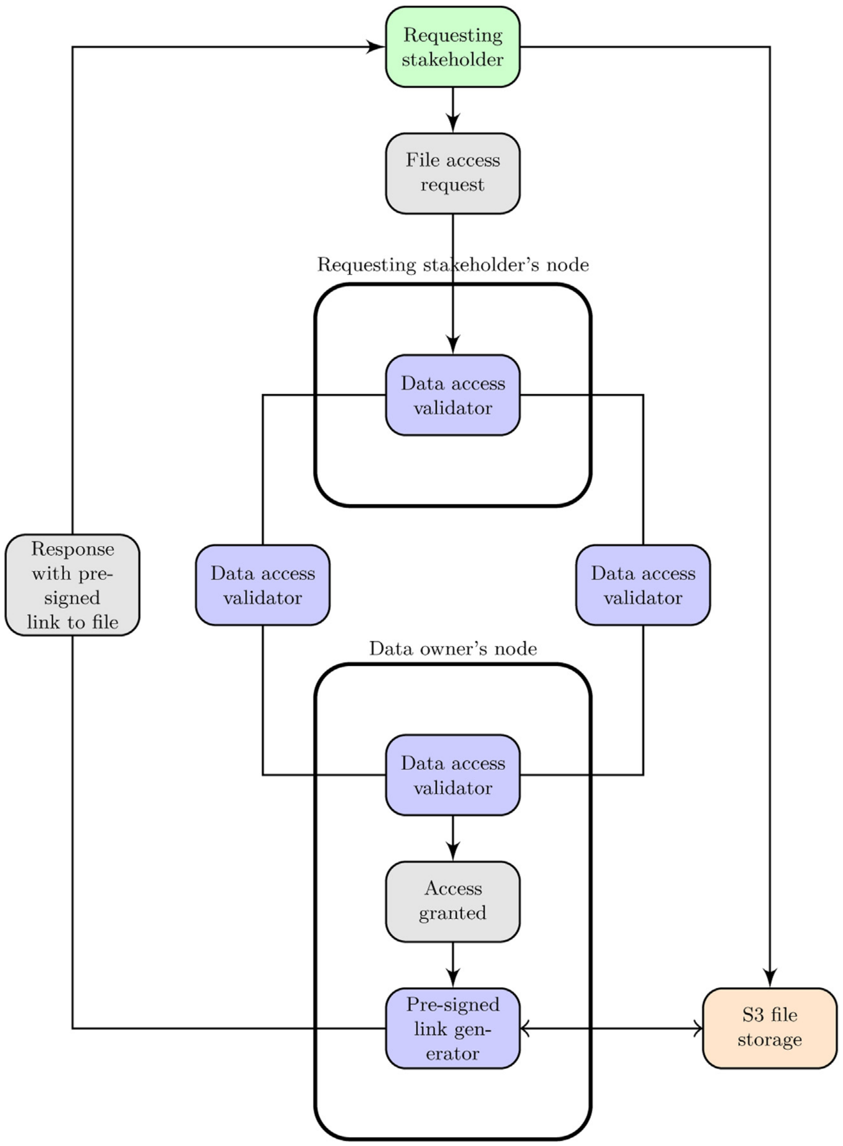


Fig. 8. Off-chain data storage.

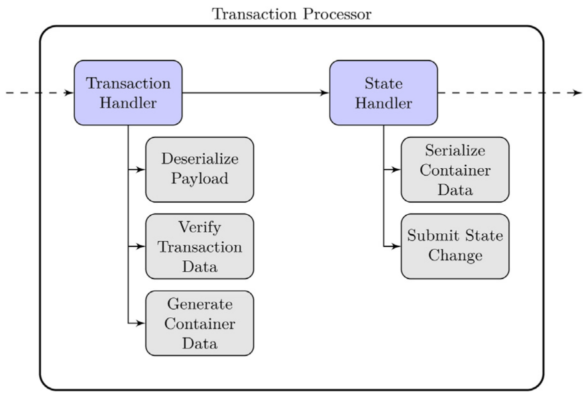


Fig. 9. Transaction processor layout in Hyperledger Sawtooth.

to log a violation of a leasing contract's operational restrictions. It is also possible to write to multiple fields and data models in the state handler. For example, when a new flight is added, the state handler adds the flight to the aircraft's flight log, but also updates the flight cycle entries.

The verification of transaction data can impose a significant load on the node, especially when large databases and/or external data sources are involved. It becomes apparent that a well-designed TP can have a big impact on overall node performance. It should be noted that:

- Many utility functions will likely be called very frequently.
- The TP handler functions will be executed repeatedly and on multiple nodes for every submitted transaction to ensure that the function is indeed deterministic.

3.2.6. *REST API & client*

The REST API provides an interface to communicate with the blockchain. It serializes the transaction payload before forwarding it to the node. The protocol buffers used to serialize the flight insertion payload are shown as an example in Code Block 1.5. The request also includes inputs and outputs fields that define the addresses that the TP is allowed to read from and write to.

It is possible to include generic addresses in those fields. This can be useful for transactions where the address cannot be fetched or calculated by the API, but it can lead to drastic performance decreases if the data model has a high transaction throughput. This

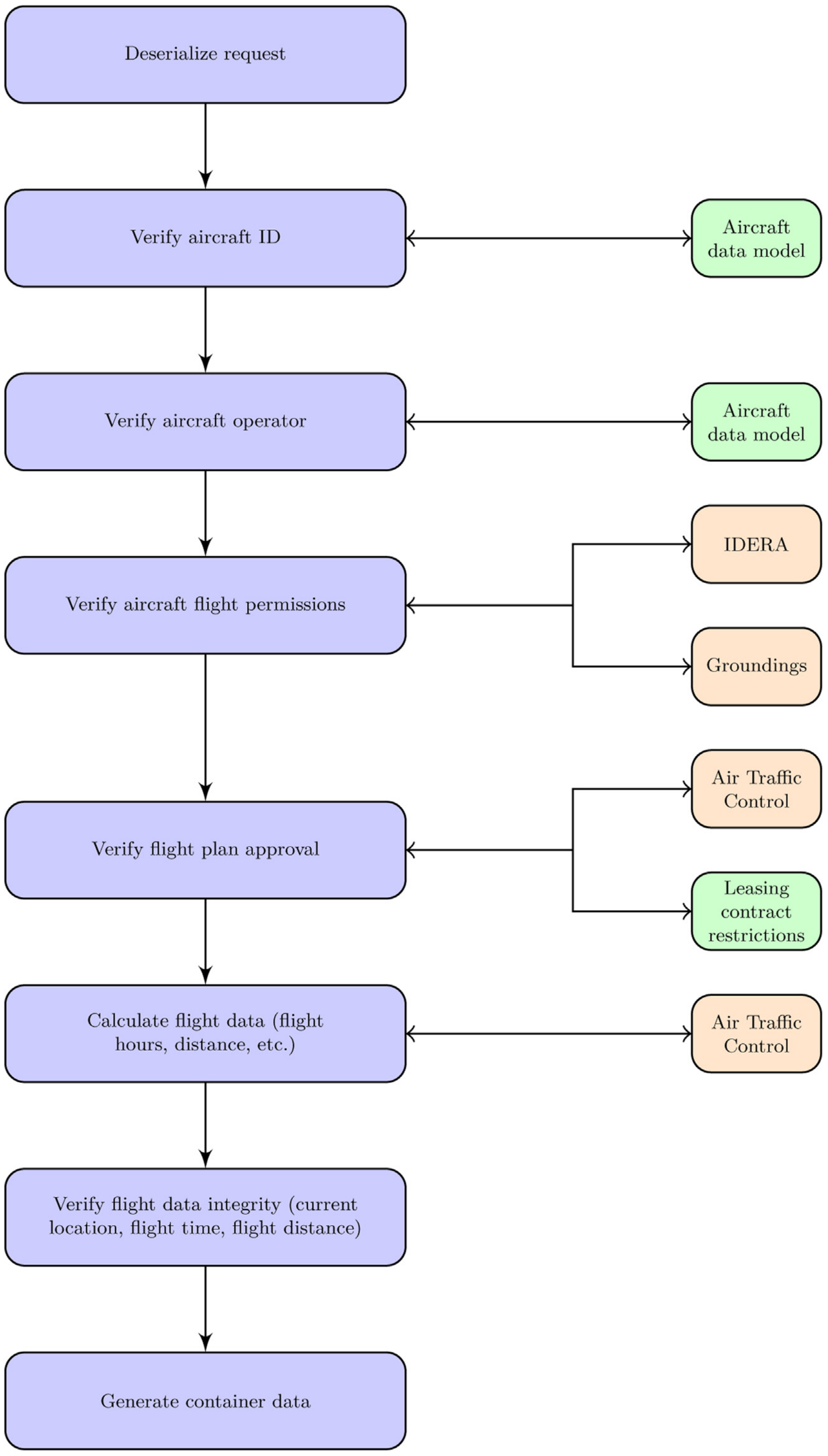

Deserialize request
Verify aircraft ID
Aircraft data model
Verify aircraft operator
Aircraft data model
Verify aircraft flight permissions
IDERA
Groundings
Verify flight plan approval
Air Traffic Control
Leasing contract restrictions
Calculate flight data (flight hours, distance, etc.)
Air Traffic Control
Verify flight data integrity (current location, flight time, flight distance)
Generate container data

Fig. 10. TP flowchart for flight data insertion. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

is due to the fact that the TP cannot schedule parallel transactions for any addresses that are currently in use.

The client that was developed for this paper is a generic implementation for human end users. It operates independently from the node and can be set up according to the customer's needs. An overview of the different components of this client can be seen in Fig. 11. A subscriber service listens to the REST API and writes new state changes to a PostgreSQL database. This design allows for quick access times to blockchain data, since the frontend app never needs to query the blockchain's REST API directly. The subscriber can make a catch-up request to the REST API in case it is missing blocks, e.g. when a new client needs to be synchronized.

The synchronized data models can be customized and enriched with data from other sources. A sample dashboard page is shown in Fig. 12. It lists the aircraft's stakeholders and their blockchain addresses together with a visual representation of the aircraft's flight cycle/flight hours history and information about its last flight.

The demo application also includes the following detail pages:

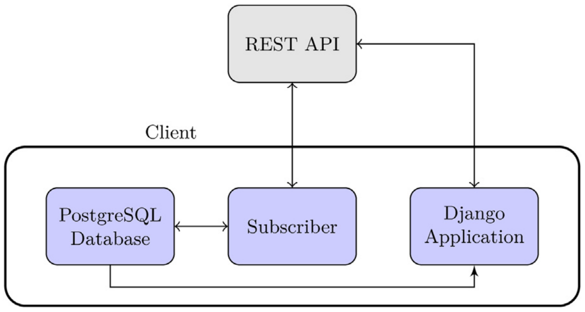


Fig. 11. Client architecture.

- The aircraft's flight log (Fig. 13).
- An AMP page with a list of tasks, events, MRO shop visits and compliance entries (Fig. 14).
- The aircraft's equipment history (Fig. 15).
- A leasing contract page with information about the delivery and redelivery, payments (Fig. 16) and compliance (Fig. 17).

3.3. Discussion

In order to validate the functional and non-functional requirements, we implemented the proposal presented in Section 3 in Python as a proof-of-concept and simulated several aircraft leases.

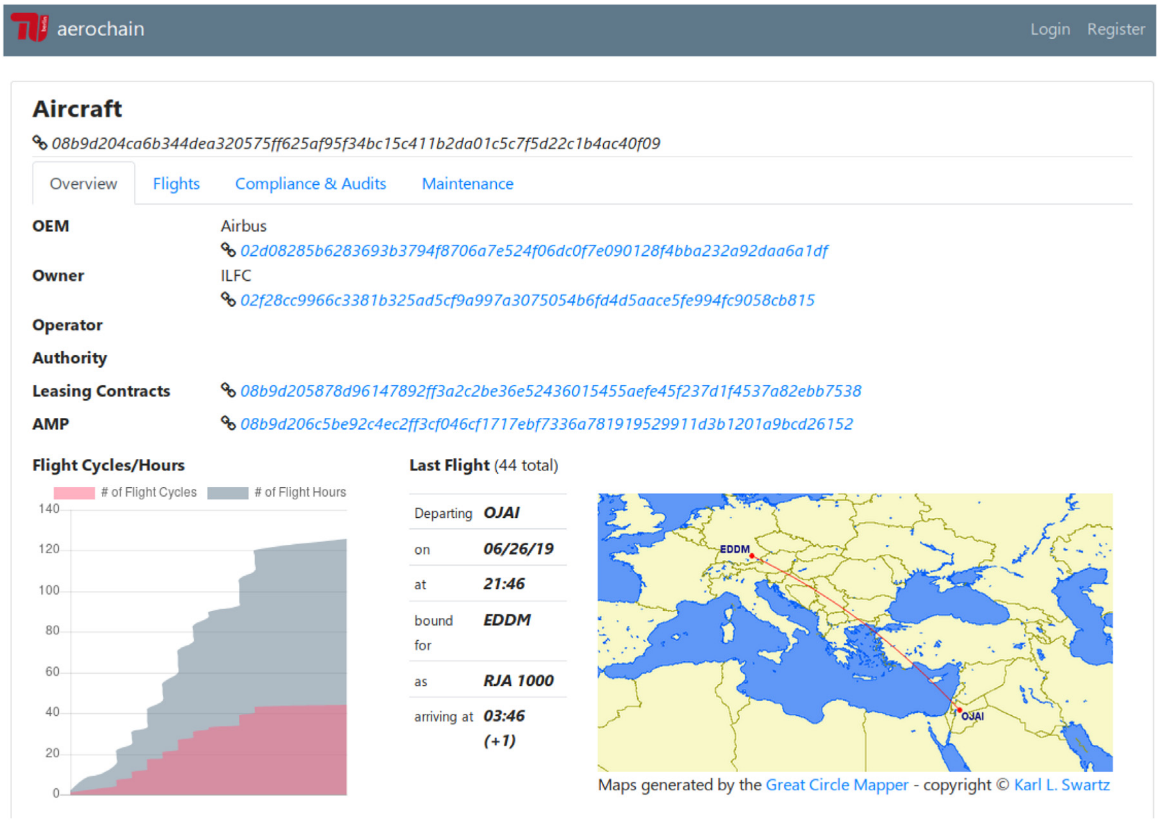
In order to ensure the accuracy of the simulation, the workload was generated using actual flight data and a complete MX task list taken from a real MPD. The flight data includes all commercial flights operated in 2018 by Finnair's and Royal Jordanian's Airbus A320 family fleet. These airlines were chosen because their combined fleet size of 50 aircraft is large enough to conduct scalability tests and because the two airlines operate in different jurisdictions. This is required for certain functional tests like transferring an aircraft to an operator that is regulated by a different authority.

The simulation has shown that the proposed design fulfils the functional requirements defined in Table 1.

One major downside of Hyperledger Sawtooth's architecture is that transactions cannot be scheduled on the blockchain itself, as the platform does not have a central clock and would require a consensus algorithm to determine the official system time. This means that scheduled transactions always have to be invoked specifically by a peer. Regular tasks and functions regarding items with an expiration date or a due date (like ADs) have to be scheduled client-side.

This can also cause issues with transaction handler functions whose outputs depend on the current time. Transactions that are submitted close to the cutoff time may be declined because of determinism failures, as system times on different nodes can vary.

On-chain hooks or signals would be a very useful feature for transactions that are triggered by other events. For instance, whenever a new flight is added to the flight log, the aircraft's AMP status should be recalculated, given that many intervals are measured in flight hours or flight cycles. It is possible to listen to these events on individual nodes and then invoke the required function, but this can cause the same transaction to be sent by multiple stakeholders. Transactions that check and update the status of a data model are particularly prone to this (for example AMP tasks that depend on the number of flight cycles/hours). Thus, the transaction would be executed more than once. While this is not a problem in terms of data integrity or security, it does use resources and will block the involved addresses from being written to while the TP is executing the transaction. An on-chain event listener for the AMP model would be a more efficient solution, but this is not possible as of today.



Aircraft

08b9d204ca6b344dea320575ff625af95f34bc15c411b2da01c5c7f5d22c1b4ac40f09

OEM	Airbus 02d08285b6283693b3794f8706a7e524f06dc0f7e090128f4bba232a92daa6a1df
Owner	ILFC 02f28cc9966c3381b325ad5cf9a997a3075054b6fd4d5aace5fe994fc9058cb815
Operator	
Authority	
Leasing Contracts	08b9d205878d96147892ff3a2c2be36e52436015455aefe45f237d1f4537a82ebb7538
AMP	08b9d206c5be92c4ec2ff3cf046cf1717ebf7336a781919529911d3b1201a9bcd26152

Flight Cycles/Hours

Last Flight (44 total)

Departing	*OJAI*
on	*06/26/19*
at	*21:46*
bound for	*EDDM*
as	*RJA 1000*
arriving at	*03:46 (+1)*

Maps generated by the Great Circle Mapper - copyright © Karl L. Swartz

Fig. 12. The Django client showing the aircraft overview page.

Given that the client converts blockchain data to a relational database schema, the application fulfils **C-1**. However, the client synchronization appears to be a bottleneck, as the network can process transactions faster than the client's event listener and database can process them. In a production setup, the event listener's traditional ETL (Extract, Transform, and Load) process should be replaced by a streaming ETL process. Possible architectures for this type of data ingestion are discussed in Meehan et al. (2017).

The GUI shows that it is possible to present a complex blockchain-based system like a web 2.0 application, as required by **E-5**. Further improvements could be made by exposing the node's interfaces through a JavaScript API similar to Ethereum's web3.js. This way, a user could send transactions like audits and file access requests directly from the GUI without having both the request and the response go through the client's backend API.

The compliance requirements in Table 3 could also be validated in the simulation. The blockchain's data models securely store the documentation listed in **B-3**. Because the aircraft's data model does not change, new stakeholders automatically have access to all relevant data (**B-2**). With a fully decentralized file storage system, the requirement to store backups in different geographic locations (**B-1**) will be fulfilled automatically. The current design still requires the stakeholder to implement a compliant backup system. Moreover, data transferability will be automatic, whereas this proof-of-concept uses a TP function that transfers the data to the new stakeholder's server.

As part of the simulation, we measured the blockchain's performance following the guidelines established by *The Hyperledger Project* (The Hyperledger Project, 2018). A total of 10 nodes were simulated in virtual environments running on a server with the following specifications:

- *CPU:* Intel® Xeon® E5-2620 v4
- *RAM:* 128GB
- *Storage:* 128TB SAS3-SSD

The blockchain reached a throughput of 1.68 transactions per second with a maximum latency of 38.67 seconds. It should be noted that the main objective of the simulation was to validate the functionalities of the TPs. While the data access patterns are realistic, the network setup of a real-world implementation (e.g. geographic distribution of nodes) has to be taken into account when simulating a production environment. Also, the simulation was capped at 50 aircraft, but sped up to simulate a full leasing cycle. Therefore, the simulation cannot be viewed as a complete performance test. However, the latency spikes observed in the simulation indicate that other publications that have reported throughputs of 300 (Pongnumkul et al., 2017) or even 700 transactions per second (Dang et al., 2019) seem to arrive at these numbers at least in part because their transaction handlers contain very little or simple business logic.

The maximum latency observed in the simulation is acceptable (**C-3**). Further work is needed to accurately measure the maximum throughput and the hardware setup necessary to achieve it (the throughput is limited by the slowest node and its CPU in particular (Pongnumkul et al., 2017)). Along the same lines, an estimation of the energy and environmental footprint of the blockchain compared to the traditional paper-based workflow could provide incentives to adopt the platform.

Additional incentives could include a modular standard library with transaction family templates and interfaces to industry-standard software. The solution could also be expanded to include more stakeholders like insurance providers, Air Traffic Control and financial institutions (or decentralized payment processors) natively instead of relying on them as external data sources.

Certain external data providers that have no stake in the blockchain itself could potentially be added to the blockchain as self-updating data models. This could reduce transaction processing times, particularly for those transaction types where network latency is the biggest bottleneck or the external data source is slower than the blockchain itself. While this strategy does not work for very dynamic data like live updates from Air Traffic Control, we have identified some candidates: Aeronautical information is updated in 28 day intervals (AIRAC cycles as defined in International Civil Aviation Organization (2013) Chapter 6) and all changes are announced at least 42 days in advance. This means that the majority of information needed to calculate flight metrics (airport codes, waypoints, etc.) could be stored on the blockchain.

This problem could be used to study the performance implications of external data sources. Particularly valuable conclusions could be drawn regarding the circumstances under which it makes sense to move data to the blockchain and how they can be parametrized (e.g. network latency, time and space complexity of the queries, etc.).

Lastly, while Hyperledger Sawtooth brings a lot of flexibility out of the box, it would be helpful to analyze the performance benefit of compiled and/or statically typed languages like Go in the TP.

4. Conclusions

Blockchain offers a set of functionality that can lead to disruptive changes in many industrial sectors, as it is the case with commercial aircraft leasing. This paper analyzes in detail the shortcomings of operational processes and methods in such an industry. Based upon this analysis, it is conducted a mapping between the needs in commercial aircraft leasing and blockchain capabilities. Along the paper, it is shown how automation and workload reduction can be bolstered with the help and guidance of an adequate blockchain arquitecture. For completeness, this roadmap is applied to the design and implementation of a decentralized solution for aircraft leasing.

We have arrived at a proof of concept, but further work is necessary to fully leverage the opportunities of decentralization in each area of this problem. First of all, it is necessary to further explore the convenience of adopting decentralized storage systems for assets management. This analysis should take into account that a centralized approach to file storage may be a necessary intermediate step on the way to a fully decentralized solution. This is due to the fact that decentralized data storage, especially across multiple jurisdictions, still suffers from a lack of legal frameworks and reference cases (Portal, 2020). For instance, an analysis of the legal situation in Germany (Hein et al., 2018) highlights aspects of contract law, liability issues, data protection and criminal law as some of the challenges. It is therefore helpful to show that this concept can be implemented even without decentralized document storage.

The addition of an identity and reputation management system compliant with self-sovereign identity (SSI) principles would further increase traceability and transparency in a way that respects individual liberties and privacy rights. Blockchain-based SSI implementations are discussed in (Ferdous et al., 2019; Baars, 2016), and driven at European level by the European Blockchain Services infrastructure (EBSI). In this context, our solution could also be integrated into a public blockchain platform such as for SSI, notarization, certifications and other legal matters, especially in a business-to-government context such as civil aviation regulatory oversight (Portal, 2020).

While not considered in current civil aviation legislation, machine-to-machine interactions are a central pillar of IoT systems and can potentially be beneficial in aircraft lifecycle management. Many processes that involve multiple parties like flight logging and payments are still initiated by human stakeholders, but a decentralized environment would enable the involved machines to carry out these tasks without human input.

This paper demonstrates that the aircraft leasing and lifecycle management sector stands to benefit from a decentralized approach to asset management, which can realistically be handled by a blockchain-based solution. It is clear that a decentralized aircraft leasing system integrates well with other decentralized applications (identity management, payments, supply chains, insurance, etc.) and will deliver more value in a more developed ecosystem. More and more companies in the aviation industry have included decentralized solutions in their IoT strategy, and it is only a matter of time until the first blockchain-based aircraft lease agreements are going to appear.

Authors' contributions

Paul Kuhle: Conceptualization, Methodology, Investigation, Software, Visualization, Writing – Original Draft, Writing – Review & Editing.

David Arroyo: Conceptualization, Methodology, Investigation, Writing – Original Draft, Writing – Review & Editing.

Eric Schuster: Conceptualization, Writing – Original Draft, Writing – Review & Editing. All authors have read and agreed to the published version of the manuscript.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgements

This work has been supported by the Comunidad de Madrid (Spain)under the project CYNAMON (P2018/TCS-4566), co-financed with FSE andFEDER EU funds, by Ministerio de Economía, Industria y Competitividad (MINECO), Agencia Estatal de Investigación (AEI), and European Regional Development Fund (ERDF, EU), through project COPCIS, grant number TIN2017-84844-C2-1-R, and by the Consejo Superior de Investigaciones Científicas (CSIC) under the project LINKA20216 ("Advancing in cybersecurity technologies", i-LINK+ program). We thank flightera.net for providing us with flight data for the simulation.

Appendix A. Different blockchain types

Table 2
A comparison of different blockchain types. Adapted and modified from (Tamayo, 2016).

	Public	Private and Consortium
	No centralized management	Single Organization (Private) Multiple Organizations (Consortium)
Participants	**Permissionless** • Anonymous • Could be malicious	**Permissioned** • Identified • Trusted
Consensus Mechanisms	**PoW, PoS, etc.** • High energy consumption • 51% attack • probabilistic finality	**Voting/Multi-Party** • Low energy consumption • lightweight and fast • absolute finality
Transaction Approval Speed	**Long** • Bitcoin: 10+ min, 60+ min considering finality	**Short** • 100 ms
USP and Use Cases	• disintermediation • cryptocurrencies	• reduction of transaction cost • B2B networks

Appendix B. Non-functional requirements

Table 3
Non-functional requirements: compliance.

Nr	Title	Description
B-1	Data Protection	All records have to be stored “in a manner that ensures protection from damage, alteration and theft” (M.A.714). Digital backups have to be stored in a different location.
B-2	Data Transferability	The CAMO has to ensure that all records can be transferred in case the CA activity is transferred to a different organization.
B-3	Documentation	The CAMO is required to keep aircraft and engine logbooks with flight records, CA/MX records as well as the aircraft’s build stand and equipment list, as described in M.A.305. Paragraph M.A.714 requires the CAMO to retain copies of these records “until two years after the aircraft has been permanently withdrawn from service”.

All requirements in this table are based on Commission Regulation (EU) No 1321/2014, Annex I (EU, 2014), specifically articles M.A.305, M.A.709 and M.A.714.

Table 4
Non-functional requirements: performance.

Nr	Title	Description
C-1	Scalability	With about 25,000 commercial aircraft in service as of 2017 and a forecast of 35,000 aircraft in 2027, the scalability of the platform is crucial (Cooper et al., 2017). While the performance of a blockchain is typically measured in transaction throughput and transaction latency, data storage and on-chain accounting for audits are also relevant metrics in the context of this particular application. Data storage costs should be minimized, and queries should not perform significantly worse than in a system with relational databases.
C-2	Throughput	With a conservative estimate of 20 daily write requests per aircraft (flight logs, MX events, etc.) and 35,000 aircraft in service, the blockchain would need to be able to process 700,000 transactions per day (about 8 transactions per second).
C-3	Latency	As opposed to many blockchain applications in the financial sector, commercial aircraft leasing is not a low-latency application. However, transactions should be finalized within a reasonable timeframe (∼10–30 min) to allow for same-day audits.

Table 5
Non-functional requirements: business needs.

Nr	Title	Description
D-1	Data Ownership	Given the sensitive nature of the data stored on this blockchain, the individual stakeholders will want to retain ownership of their data. This means that files will either have to be stored on-chain or on a server of the data owner’s choosing. Both solutions require an access control system managed by the blockchain.
D-2	Operating Costs	While enterprise blockchains are often implemented to increase efficiency and reduce costs (Al-Jaroodi and Mohamed, 2019), it is important to understand the costs associated with operating a large-scale blockchain. Particularly, the energy footprint of blockchains has been the subject of debate (Vranken, 2017) and should be taken into consideration.

Table 6
Non-functional requirements: usability.

Nr	Title	Description
E-1	Integration	The platform will be integrated into a landscape of legacy IT solutions (e.g. MRO software). Therefore, the solution needs to integrate well into legacy systems, but emphasize and support an increasingly decentralized architecture (e.g. decentralized payments and supply chain tracking).
E-2	Flexibility	Given the different business needs of the involved stakeholders, each party may wish to customize their blockchain node (e.g. for data synchronization, interfaces to other systems). The architecture of the blockchain platform should avoid vendor lock-in, specifically with regards to hosting services and programming languages.
E-3	Ecosystem	Smaller stakeholders may not have the resources to maintain their own nodes or to develop custom software to integrate them into their existing IT infrastructure. Therefore, the solution should facilitate the development of an ecosystem around the blockchain platform. This ecosystem would enable third parties to offer solutions like managed hosting and custom blockchain clients.
E-4	Maintainability	Given the long time horizon of this project, a focus on maintainability and support is essential. The underlying blockchain technology and the libraries used for core functionalities should be mature and offer long-term enterprise support.
E-5	Interface Design	The end user should not need blockchain-specific knowledge or training to be able to use the platform. GUIs for humans should behave like web 2.0 applications. Other systems should be able to communicate with the blockchain system using established industry standards like REST APIs.

Appendix C. Data models

Listing 1.1: Data model for stakeholders

```
message Owner {
    // The owner's unique public key
    string public_key = 1;

    // A human-readable name identifying the owner
    string name = 2;

    // A human-readable name identifying the country
    string country = 3;

    // Approximately when the owner was registered,
         as a Unix UTC timestamp
    uint64 timestamp = 4;
}

message OwnerContainer {
    repeated Owner entries = 1;
}
```

Listing 1.2: Data model for aircraft

```
message Aircraft {
  // ==============================================
  // =         S T A K E H O L D E R S            =
  // ==============================================

  // OEM (Manufacturer)
  // ===========================
  message OEM {
      // Public key of the OEM
      string oem_id = 1;

      // Approximately when the OEM was updated,
      // as a Unix UTC timestamp
      uint64 timestamp = 2;
  }

  repeated OEM oems = 1;

  // Owner
  // ===========================
  message Owner {
      // Public key of the owner who owns the
            aircraft
      string public_key = 1;

      // Approximately when the owner was updated,
      // as a Unix UTC timestamp
      uint64 timestamp = 2;
  }

  repeated Owner owners = 2;

  // Operator/CAMO
  // ===========================
  message Operator {
      // Public key of the operator who
      // operates the aircraft
      string public_key = 1;

      // Approximately when the owner was updated,
      // as a Unix UTC timestamp
      uint64 timestamp = 2;
  }

  repeated Operator operators = 3;

  // Authority
  // ===========================
  message Authority {
      // Public key of the responsible authority
      string public_key = 1;

      // Aircraft tailnumber
      string registration = 2;

      // Approximately when the owner was updated,
      // as a Unix UTC timestamp
      uint64 timestamp = 3;
  }

  repeated Authority authorities = 4;

  // ==============================
  // =   B A S I C    D A T A     =
  // ==============================

  message BasicData {

      // Aircraft Type
      // (Name of aircraft as defined by OEM,
      // e.g. Airbus A320-200)
      string type = 1;

      // Additional type spec as defined by OEM,
      // e.g. A320-231
      string subtype = 2;

      // engine type installed on aircraft
      string engine_type = 3;

      // APU type installed on aircraft
      string apu_type = 4;

      // Manufacturer Serial Number
      string msn = 5;

      // Weight Variant
      string weight_variant = 6;

      // a Unix UTC timestamp
      uint64 timestamp = 7;
  }

  BasicData basicdata = 5;

  // ==========================================
  // =     C U R R E N T    S T A T U S       =
  // ==========================================

  // Current Location of A/C
  // ===========================
  message Location {
      // ICAO code of location
      string icao_id = 1;

      // Approximately when the aircraft arrived at
      // its current location, as a Unix UTC
            timestamp
      uint64 timestamp = 2;
  }

  repeated Location locations = 6;

  // Number of Hrs/Cycles
  // ===========================
  message HoursCycles {
      // current number of flight cycles
      uint64 cycles = 1;

      // current number of flight hours
      uint64 hours = 2;

      // Approximately when the information
      // was updated, as a Unix UTC timestamp
      uint64 timestamp = 3;
  }

  repeated HoursCycles hour_cycle_history = 7;

  // Legal Status
  // ===========================
  message LegalStatus {
      // if aircraft is grounded,
      // flag is set to 1, else 0
      uint64 grounded = 1;

      // IDs of overdue ADs
      repeated string overdue_ad = 2;

      repeated string delayed_ad = 3;

      // Task No of overdue AMP tasks
      repeated string overdue_amp = 4;

      repeated string delayed_amp = 5;

      // etops 0 or 1
      uint64 etops = 6;

      // Approximately when the information
      // was updated, as a Unix UTC timestamp
      uint64 timestamp = 7;
  }

  repeated LegalStatus legal_status_history = 8;

  // ==============================================
  // =          M A I N T E N A N C E             =
  // ==============================================

  // Service Bulletins/Mods
  // ===========================
  message Mod {
      // public key of SB
      string sb_key = 1;

      // Mod ID
      string mod_id = 2;

      // a Unix UTC timestamp of when it was added
      uint64 timestamp_added = 3;

      // a Unix UTC timestamp of when it was
            installed
      uint64 timestamp_installed = 4;

      // a Unix UTC timestamp of when it was
            installed
      string event_reference = 5;
  }

  // Mods
  // ===========================
  repeated Mod mods = 9;


  // Airworthiness Directives
  // ===========================
  message AD {
      // public key of AD
      string ad_key = 1;

      // when is AD due
      uint64 timestamp_due_date = 2;

      // a Unix UTC timestamp of when it was added
      uint64 timestamp_added = 3;

      // a Unix UTC timestamp of when it was
            installed
      uint64 timestamp_processed = 4;

      // a Unix UTC timestamp of when it was
            installed
      string event_reference = 5;
  }

  // A/Ds
  // ===========================
  repeated AD ads = 10;

  // Last MRO event
  // ===========================
  message ShopVisit {
      // event_id
      string event_id =1;

      // Type of visit (scheduled, unplanned)
      string type = 2;

      // Responsible Person CAMO
      string responsibleCamo = 3;

      // MRO Shop
      string shop = 4;

      // reference to AD
      string reference_ad = 5;

      // reference to SB
      string reference_sb = 6;

      // reference to MPD task
      string reference_mpd = 7;

      // link to documentation
      string documentation_link = 8;

      // hash of document
      string documentation_hash = 9;

      // timestamp of when work was finished
      uint64 timestamp_workdone = 10;

      // Approximately when the information was
      // updated, as a Unix UTC timestamp
      uint64 timestamp = 11;
  }

  repeated ShopVisit shop_visits = 11;

  // equipment list
  message Equipment {
      // equipment id
      string equipment_id = 1;

      // equipment oem
      string oem = 2;

      // equipment name
      string name = 3;

      // serial number
      string msn = 4;

      // maintenance event reference
      string event_id = 5;

      // type of event (installation/removal)
      string event_type = 6;

      // voluntary or mandatory equipment change
      string type_of_change = 7;

      // timestamp of entry
      uint64 timestamp = 8;
  }
  repeated Equipment equipment_list = 12;

  // ==============================================
  // =            O P E R A T I O N S             =
  // ==============================================

  // Last recorded flight
  // ===========================
  message Flight {
      // ICAO Code of departure point
      string from_icao_id = 1;

      // ICAO Code of arrival point
      string to_icao_id = 2;

      // departure time as a Unix UTC timestamp
      uint64 departure_time = 3;

      // arrival time as a Unix UTC timestamp
      uint64 arrival_time = 4;

      // travelled distance in nautical miles
      uint64 distance = 5;

      // Type of flight
      // (ferry, passenger, test, training)
      string type = 6;

      // Flight number, e.g. LH1800
      string operator_id = 7;

      // Flight number, e.g. LH1800
      uint64 flightno = 8;

      // Approximately when the location was updated
            ,
      // as a Unix UTC timestamp
      uint64 timestamp = 9;
  }

  repeated Flight flight_history = 13;

  string aircraft_id = 14;

   repeated string associated_leasing_contracts = 15;

   repeated string associated_maintenance_plans = 16;

 }

message AircraftContainer {
    repeated Aircraft entries = 1;
}
```

Listing 1.3: Data model for AMPs

```
message ApprovedMaintenancePlan {

    // ID of authority that accepted this AMP
    string authority_id = 1;

    // ID of operator who accepted this AMP
    string operator_id = 2;

    // ID of involved aircraft
    string aircraft_id = 3;

    // status of AMP

    message Status {
        // status (is AMP valid or void)
        string status = 1;

        // timestamp
        uint64 timestamp = 2;
    }

    repeated Status status = 4;

    // maintenance tasks
    message Task {
        // task number
        string task_no = 1;

        // task source (e.g. MRB Report)
        string source = 2;

        // description of task
        string description = 3;

        // zones (e.g. 152, 153)
        repeated string zones = 4;

        // reference for further documentation
        string reference = 5;

        message ThresholdIntervall {
            // amount of threshold or interval
            uint64 amount = 1;

            // maximum variation of threshold or
                  interval
            uint64 max_var = 2;

            // unit of amount and max_var (can be
                  flight hours, cycles or seconds)
            string unit = 3;
        }

        // options for threshold (first maintenance)
              , e.g.
        // 25000 cycles or 6 years, whichever comes
              first
        repeated ThresholdIntervall
              threshold_options = 6;

        // options for interval (after first
              maintenance), e.g.
        // 25000 cycles or 6 years, whichever comes
              first
        repeated ThresholdIntervall interval_options
               = 7;

        // task applies to following aircraft
        message Applicability {
            // Aircraft type
            string type = 1;

            // Aircraft subtype
            string subtype = 2;

            // Etops build stand of aircraft
            string etops = 3;

       repeated uint64 operator_signatures = 7;

       uint64 timestamp = 8;

       string amp_id = 9;

       // sequence of AMP for specific
       // operator-aircraft-authority-combo
       // (i.e. 5th AMP)
```

```
         string faa = 4;

         // aircraft engine type
         string engine = 5;

         // weight variants
         repeated string weight_variants = 6;

         // task applies to aircraft with or
               without certain mods
         message Mods {
             // 'pre' for aircraft without mods
             // 'post' for aircraft with mods
             string prepost = 1;

             // Mod ID
             string mod = 2;

             // List of applicable service
                   bulletins
             repeated string sb = 3;

             // Mods with same comboNo have 'and'
                    logic (i.e. all/neither),
             // Mods with different comboNo have
                   'or' logic (i.e. either/or)
             string combono = 4;
         }

         repeated Mods mods = 7;

         // unix timestamps of when applicability
                was added or removed
         uint64 start_applicability = 8;

         uint64 end_applicability = 9;
     }

     repeated Applicability applicabilty = 8;

     // 0 or 1
     uint64 is_applicable = 9;

     // maintenance events
     message Event {
         // when was the event
         uint64 event_timestamp = 1;

         // event id to cross ref aircraft
               address
         string event_id = 2;

         // total flight hours at event
         uint64 hours_at_event = 3;

         // total flight cycles at event
         uint64 cycles_at_event = 4;

         // hours or cycles of delay
         uint64 maintenance_delay = 5;

         // hours or cycles
         string maintenance_delay_type = 6;

         // delay within maximum variation
         uint64 delay_within_max_var = 7;
     }

     repeated Event maintenance_events = 10;

     uint64 timestamp = 11;

 }
 repeated Task tasks = 5;

 // contract signatures

 repeated uint64 authority_signatures = 6;

    uint64 sequence = 10;

}

message ApprovedMaintenancePlanContainer {
    repeated ApprovedMaintenancePlan entries = 1;
}
```

Listing 1.4: Data model for leasing contracts

```
message LeasingContract {

    // ==================================
    // =      S T A K E H O L D E R S     =
    // ==================================

    // aircraft ID of involved aircraft

    string aircraft_id = 1;

    // involved parties: Lessor and Lessee

    string lessor_id = 2;

    string lessee_id = 3;

    // full contract

    message Contract {
        // link to contract
        string link = 1;

        // hash
        string hash = 2;

        // status (is contract valid or void)
        string status = 3;

        // timestamp
        uint64 timestamp = 4;
    }

    repeated Contract contract = 4;

    // initial equipment list
    message Equipment {
        // equipment id
        string equipment_id = 1;

        // equipment oem
        string oem = 2;

        // equipment name
        string name = 3;

        // serial number
        string msn = 4;
    }

    repeated Equipment initial_equipment_list = 5;

    // delivery

    message Delivery {

        // ICAO code of delivery location
        string delivery_location = 1;

        // delivery date as a UNIX timestamp
        uint64 delivery_date = 2;

        // excusable delivery delay in seconds
        uint64 excusable_delay = 3;

        // status (not yet delivered, delivered,
              accepted)
        string delivery_status = 4;

        // delay of delivery (!<excusable_delay,
        // otherwise lessee has option to void
              contract)
        uint64 delivery_delay = 5;

        // link to certificate of acceptance
        string acceptance_link = 6;

       // hash of certificate of acceptance
       string acceptance_hash = 8;

       // link to certificate of delivery condition
       string condition_link = 9;

       // hash of certificate of delivery condition
       string condition_hash = 10;

       // status can only be set to accepted if the
              following conditions are met:
       // -no outstanding AMP items
       // -no outstanding ADs
       // -removed voluntary equipment changes
       // -compliance with SBs
       // The first two conditions ahve to be met
             under all circumstances,
       // the last two can be manually cancelled by
              the lessor
       uint64 aircraft_returnable = 11;

       // timestamp
       uint64 timestamp = 12;
   }

   Redelivery redelivery = 8;

   // payments

   message Payments {
       // basic amount due per rental period in USD
       uint64 basic_rent_amount = 1;

       // additional rent adjustments in percent
       uint64 add_rent_adjustment = 2;

       // additional rent of X USD per Flight Hour
             or Cycle flown by
       // the Airframe, whichever is greater
       uint64 airframe_add_rent_amount = 3;

       // additional rent of X USD per Flight Hour
             or Cycle flown by
       // the engine, whichever is greater
       uint64 engine_add_rent_amount = 4;

       // payment for LLP of X USD per Cycle flown
             by
       // the engine
       uint64 engine_llp_rent_amount = 5;

       // additional rent for landing gear of X USD
              per Flight Hour flown by
       // the aircraft, whichever is greater
```

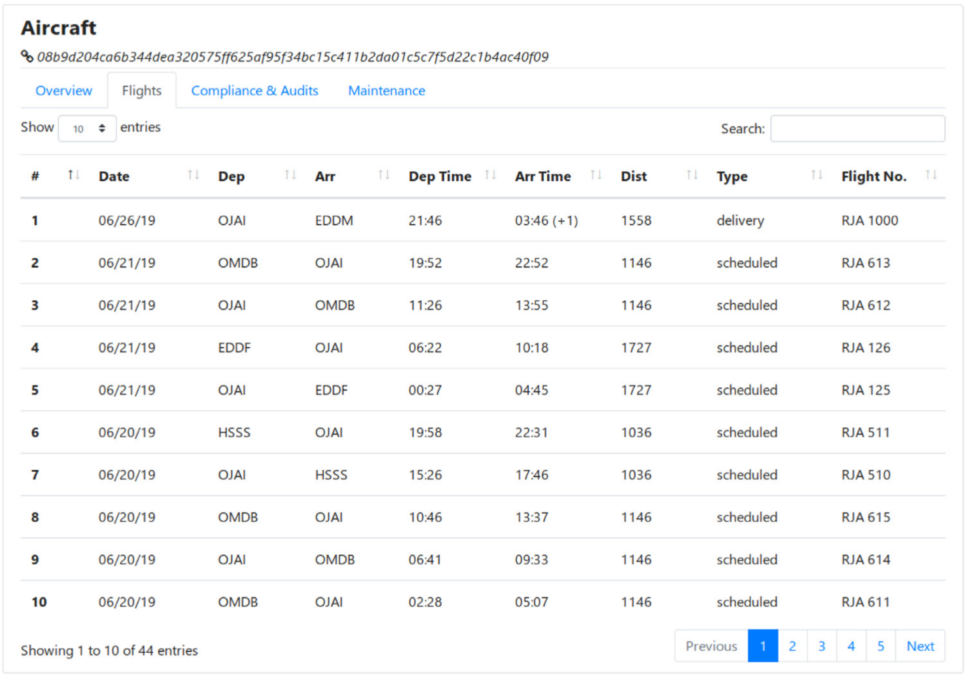
```
        // link to certificate of delivery condition
        string condition_link = 8;

        // hash of certificate of delivery condition
        string condition_hash = 9;

        // status can only be set to accepted if the
               following conditions are met:
        // -no outstanding AMP items
        // -no outstanding ADs
        uint64 aircraft_deliverable = 10;

        // timestamp
        uint64 timestamp = 11;

    }

    Delivery delivery = 6;

    // operations
    message Operation {
        // habitual base as per contract
        // state, province or country
        string habitual_base = 1;

        // habitual base as per flight ops (avg.
              over contract length
        // if contract has been active for 2 rental
              periods or longer)
        // state, province or country
        string actual_habitual_base = 2;

        // excluded countries (do not operate
              aircraft there)
        repeated string excluded_countries = 3;

        // timestamp
        uint64 timestamp = 4;
    }

    repeated Operation operations = 7;

    // redelivery

    message Redelivery {

        // ICAO code of delivery location (can be
              set during contract
        // but only in specified countries)
        string delivery_location = 1;

        // acceptable locations of delivery (
              countries)
        repeated string
              acceptable_delivery_locations = 2;

        // delivery date as a UNIX timestamp
        uint64 delivery_date = 3;

        // excusable delivery delay in seconds
        uint64 excusable_delay = 4;

        // status (not yet delivered, delivered,
              accepted)
        string delivery_status = 5;

        // delay of delivery (!<excusable_delay,
        // otherwise lessee has option to void
              contract)
        uint64 delivery_delay = 6;

        // link to certificate of acceptance
        string acceptance_link = 7;


        uint64 landing_gear_add_rent_amount = 6;

    }
    Payments payments = 9;

    message Balance {
        // amount in dollars
        uint64 amount = 1;

        // due date
        uint64 due_date = 2;

        // timestamp
        uint64 timestamp = 3;

    }
    repeated Balance current_balance = 10;

    // maintenance

    message Maintenance {
        // approved maintenance plan ID
        string amp_id = 1;

        // approved shops [list]
        repeated string asp_id = 2;

        // final maintenance performer
        string final_asp = 3;

        // timestamp
        uint64 timestamp = 4;
    }

    Maintenance maintenance = 11;

    // contract signatures timestamps

    repeated uint64 lessor_signatures = 12;

    repeated uint64 lessee_signatures = 13;

    uint64 timestamp = 14;

    string contract_id = 15;

    repeated Equipment return_equipment_list = 16;
}

message LeasingContractContainer {
    repeated LeasingContract entries = 1;
}
```

Listing 1.5: Data model for transaction payloads

```
message AeroPayload{
    enum Action {
        // ...
        INSERT_FLIGHT = 5;
        // ...
    }

    // Whether the payload contains a create owner,
           create record,
    // update record, or transfer record action
    Action action = 1;

    // The transaction handler will read from just
           one of these fields
    // according to the action

    // ...
    InsertFlightAction insert_flight = 7;
    // ...

    // Approximately when transaction was submitted,
            as a Unix UTC timestamp
    uint64 timestamp = 27;
}
```

```
    // departure airfield
    string dep_icao = 2;

    // arrival airfield
    string arr_icao = 3;

    // time of departure
    string timestamp_dep = 4;

    // time of arrival
    string timestamp_arr = 5;

    // Type of flight
    string type = 6;

    // operator_id
    string operator_id = 7;

    // flight number
    uint64 flightno = 8;
}
```

Appendix D. Django application

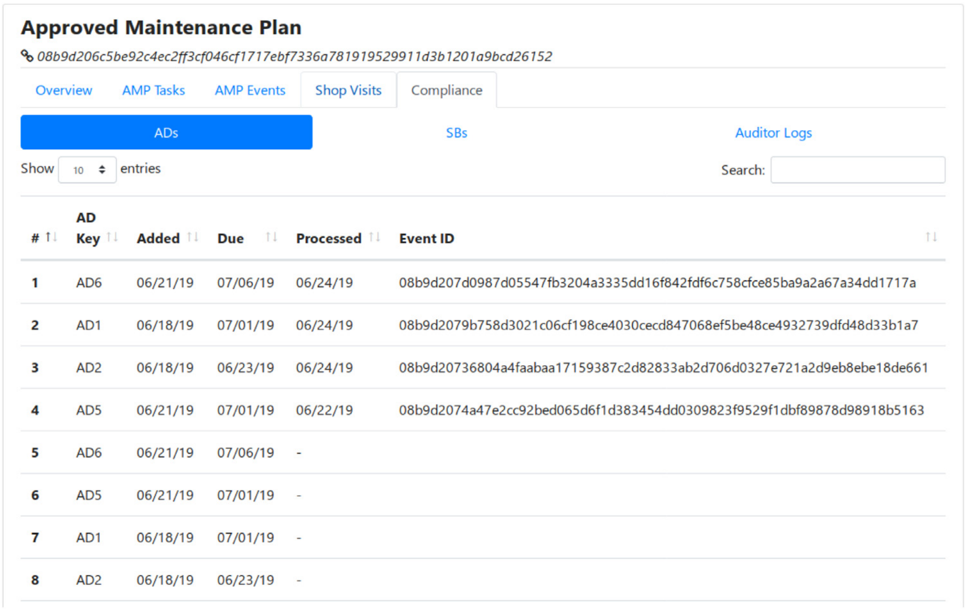

Aircraft

08b9d204ca6b344dea320575ff625af95f34bc15c411b2da01c5c7f5d22c1b4ac40f09

#	Date	Dep	Arr	Dep Time	Arr Time	Dist	Type	Flight No.
1	06/26/19	OJAI	EDDM	21:46	03:46 (+1)	1558	delivery	RJA 1000
2	06/21/19	OMDB	OJAI	19:52	22:52	1146	scheduled	RJA 613
3	06/21/19	OJAI	OMDB	11:26	13:55	1146	scheduled	RJA 612
4	06/21/19	EDDF	OJAI	06:22	10:18	1727	scheduled	RJA 126
5	06/21/19	OJAI	EDDF	00:27	04:45	1727	scheduled	RJA 125
6	06/20/19	HSSS	OJAI	19:58	22:31	1036	scheduled	RJA 511
7	06/20/19	OJAI	HSSS	15:26	17:46	1036	scheduled	RJA 510
8	06/20/19	OMDB	OJAI	10:46	13:37	1146	scheduled	RJA 615
9	06/20/19	OJAI	OMDB	06:41	09:33	1146	scheduled	RJA 614
10	06/20/19	OMDB	OJAI	02:28	05:07	1146	scheduled	RJA 611

Showing 1 to 10 of 44 entries

Fig. 13. The Django client showing the aircraft's flight logs.



Approved Maintenance Plan

08b9d206c5be92c4ec2ff3cf046cf1717ebf7336a781919529911d3b1201a9bcd26152

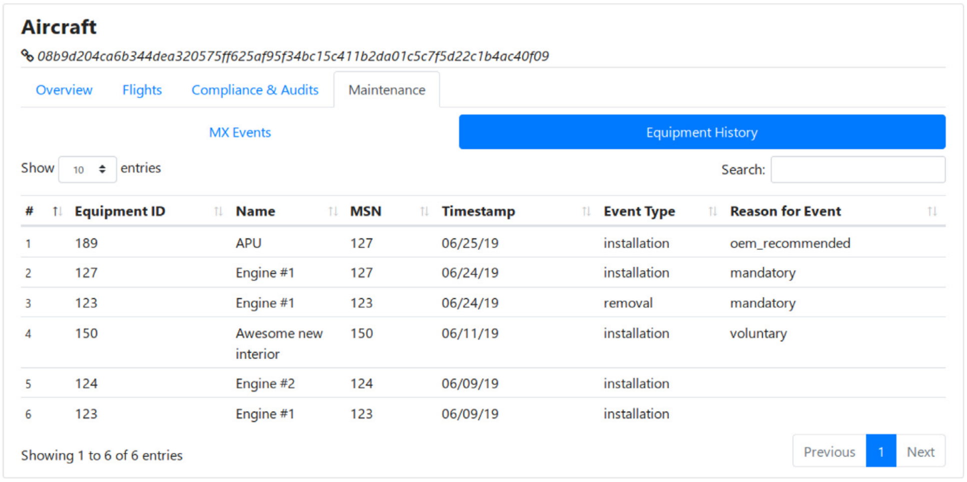
Overview | AMP Tasks | AMP Events | Shop Visits | Compliance

ADs | SBs | Auditor Logs

Show 10 entries   Search:

#	AD Key	Added	Due	Processed	Event ID
1	AD6	06/21/19	07/06/19	06/24/19	08b9d207d0987d05547fb3204a3335dd16f842fdf6c758cfce85ba9a2a67a34dd1717a
2	AD1	06/18/19	07/01/19	06/24/19	08b9d2079b758d3021c06cf198ce4030cecd847068ef5be48ce4932739dfd48d33b1a7
3	AD2	06/18/19	06/23/19	06/24/19	08b9d20736804a4faabaa17159387c2d82833ab2d706d0327e721a2d9eb8ebe18de661
4	AD5	06/21/19	07/01/19	06/22/19	08b9d2074a47e2cc92bed065d6f1d383454dd0309823f9529f1dbf89878d98918b5163
5	AD6	06/21/19	07/06/19	-	
6	AD5	06/21/19	07/01/19	-	
7	AD1	06/18/19	07/01/19	-	
8	AD2	06/18/19	06/23/19	-	

Fig. 14. The Django client showing the aircraft's AMP page.



Aircraft

08b9d204ca6b344dea320575ff625af95f34bc15c411b2da01c5c7f5d22c1b4ac40f09

Overview | Flights | Compliance & Audits | Maintenance

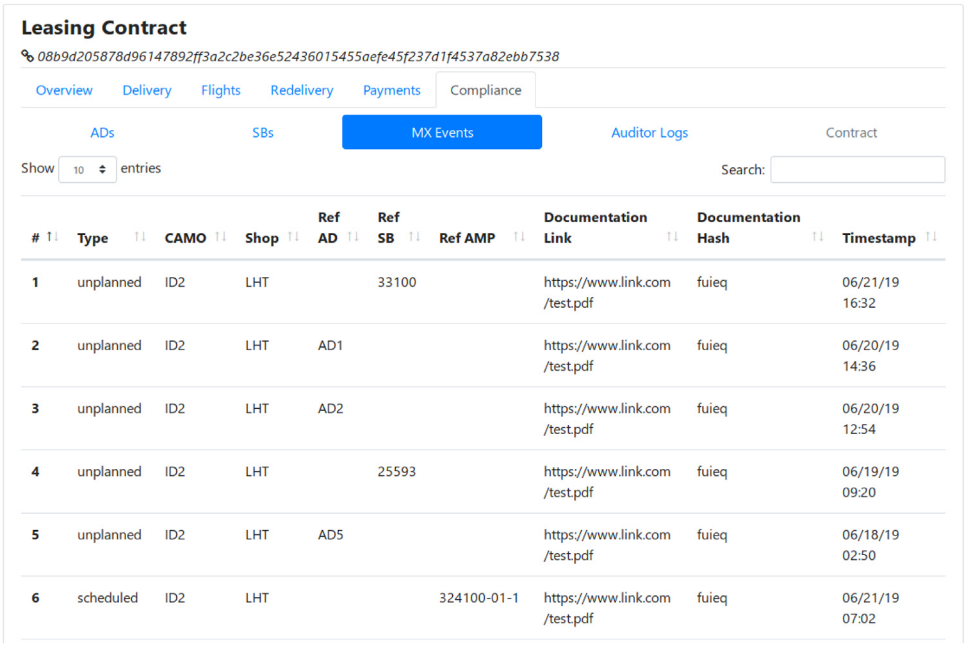
MX Events | Equipment History

Show 10 entries   Search:

#	Equipment ID	Name	MSN	Timestamp	Event Type	Reason for Event
1	189	APU	127	06/25/19	installation	oem_recommended
2	127	Engine #1	127	06/24/19	installation	mandatory
3	123	Engine #1	123	06/24/19	removal	mandatory
4	150	Awesome new interior	150	06/11/19	installation	voluntary
5	124	Engine #2	124	06/09/19	installation	
6	123	Engine #1	123	06/09/19	installation	

Showing 1 to 6 of 6 entries   Previous 1 Next

Fig. 15. The Django client showing the aircraft's equipment history page.

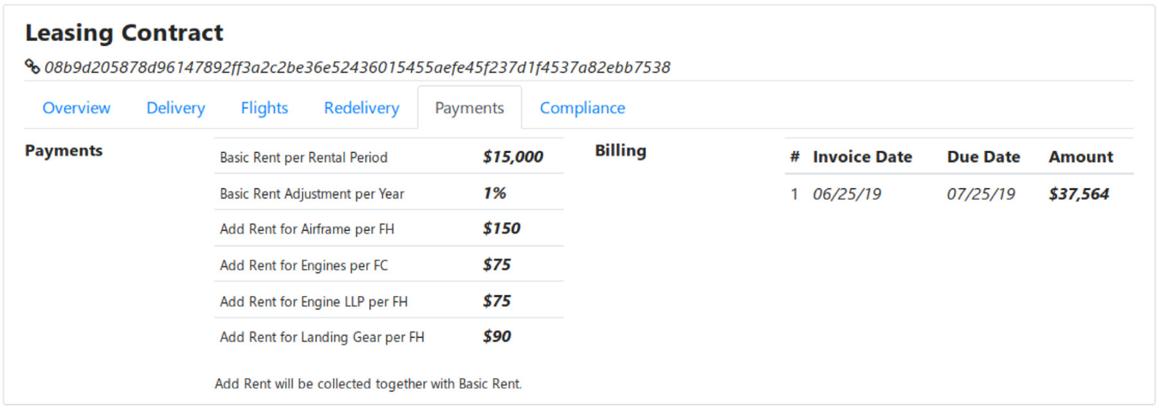


Leasing Contract

08b9d205878d96147892ff3a2c2be36e52436015455aefe45f237d1f4537a82ebb7538

#	Type	CAMO	Shop	Ref AD	Ref SB	Ref AMP	Documentation Link	Documentation Hash	Timestamp
1	unplanned	ID2	LHT		33100		https://www.link.com/test.pdf	fuieq	06/21/19 16:32
2	unplanned	ID2	LHT	AD1			https://www.link.com/test.pdf	fuieq	06/20/19 14:36
3	unplanned	ID2	LHT	AD2			https://www.link.com/test.pdf	fuieq	06/20/19 12:54
4	unplanned	ID2	LHT		25593		https://www.link.com/test.pdf	fuieq	06/19/19 09:20
5	unplanned	ID2	LHT	AD5			https://www.link.com/test.pdf	fuieq	06/18/19 02:50
6	scheduled	ID2	LHT			324100-01-1	https://www.link.com/test.pdf	fuieq	06/21/19 07:02

Fig. 16. The Django client showing the aircraft's MX history.



Leasing Contract

08b9d205878d96147892ff3a2c2be36e52436015455aefe45f237d1f4537a82ebb7538

Payments

Basic Rent per Rental Period	$15,000
Basic Rent Adjustment per Year	1%
Add Rent for Airframe per FH	$150
Add Rent for Engines per FC	$75
Add Rent for Engine LLP per FH	$75
Add Rent for Landing Gear per FH	$90

Add Rent will be collected together with Basic Rent.

Billing

#	Invoice Date	Due Date	Amount
1	06/25/19	07/25/19	$37,564

Fig. 17. The Django client showing a leasing contract's payment page.

References

Al-Jaroodi, J., Mohamed, N., 2019]. Blockchain in industries: a survey. IEEE Access 7, 36500–36515.

Arribas, I., Arroyo, D., Reshef Kera, D., 2020]. Sandbox for minimal viable governance of blockchain services and daos: Claudia. In: Prieto, J., Pinto, A., Das, A.K., Ferretti, S. (Eds.), Blockchain and Applications. Springer International Publishing, Cham, pp. 24–30.

2020]. Aviation Working Group: Global Aircraft Trading System. http://www.awg.aero/project/gats/.

Baars, D., 2016]. Towards Self-Sovereign Identity Using Blockchain Technology Master's Thesis. University of Twente.

Cooper, T., Smiley, J., Porter, C., Precourt, C., 2017]. Global Fleet & MRO Market Forecast Summary 2017–2027 (June).

Dang, H., Dinh, T.T.A., Loghin, D., Chang, E.C., Lin, Q., Ooi, B.C., 2019]. Towards scaling blockchain systems via sharding. In: Proceedings of the 2019 International Conference on Management of Data. SIGMOD'19. ACM, New York, NY, USA, pp. 123–140, http://dx.doi.org/10.1145/3299869.3319889.

Di Vaio, A., Varriale, L., 2020]. Blockchain technology in supply chain management for sustainable performance: evidence from the airport industry. Int. J. Inf. Manag. 52, 102014.

ouncil of the European Union: Regulation (european parliament and of the council of 20 february 2008 on common rules in the field of civil aviation and establishing a european aviation safety agency, and repealing council directive 91/670/eec, regulation (ec) no 1592/2002 and directive 2004/36/ec., 2008. Off. J. Eur. Union L 79/1 (March).

ouncil of the European Union: Commission regulation (. Official Journal of the European Union L 79/1 (March 2008).Council of the European Union: Commission regulation (eu) no, 1321Council of the European Union: Commission regulation (eu) no 1321/2014 of 26 november 2014 on the continuing airworthiness of aircraft and aeronautical products, parts and appliances, and on the approval of organisations and personnel involved in these tasks., 2014. Off. J. Eur. Union L 362/1 (December).

ouncil of the European Union: Regulation (e. Official Journal of the European Union L 362/1 (December 2014).Council of the European Union: Regulation (eu), 2018Council of the European Union: Regulation (eu) 2018/1139 of the european parliament and of the council of 4 european union aviation safety agency, and amending regulations (ec) no 2111/2005, (ec) no 1008/2008, (eu) no 996/2010, (eu) no 376/2014 and directives 2014/30/eu and 2014/53/eu of the european parliament and of the council, and repealing regulations (ec) no 552/2004 and (ec) no 216/2008 of the european parliament and of the council and council regulation (eec) no 3922/91., 2018. c) no, 2008Off. J. Eur. Union L 212/1 (August).

2019]. Federal Aviation Administration: FAA Proposes $715,438 Civil Penalty Against Allegiant Air. https://www.faa.gov/news/press_releases/news_story.cfm?newsId=23798.

Ferdous, M.S., Chowdhury, F., Alassafi, M.O., 2019. n search of self-sovereign identity leveraging blockchain technology. IEEE Access 7, 103059–103079.

2019]. Forum, W.E.: Inclusive Deployment of Blockchain for Supply Chains: Part 1 – Introduction, Tech. Rep. http://www3.weforum.org/docs/WEF_Introduction_to_Blockchain_for_Supply_Chains.pdf.

2019]. Google LLC: Protocol Buffers. https://developers.google.com/protocol-buffers/.

Hanley, D.P., 2011]. Aircraft Operating Leasing: A Legal and Practical Analysis in the Context of Public and Private International Air Law. Dissertation, Leiden University https://openaccess.leidenuniv.nl/handle/1887/18146.

Hein, C., Wellbrock, W., Hein, C., 2018]. Rechtliche Herausforderungen von Blockchain-Anwendungen: Straf-, Datenschutz-und Zivilrecht. Springer.

Heiss, J., Eberhardt, J., Tai, S., 2019]. From oracles to trustworthy data on-chaining systems. In: 2019 IEEE International Conference on Blockchain (Blockchain), IEEE, pp. 496–503.

Huang, H., Lin, J., Zheng, B., Zheng, Z., Bian, J., 2020. hen blockchain meets distributed file systems: An overview, challenges, and open issues. IEEE Access 8IEEE Access 8, 50574–50586.

2020]. Hyperledger: Supporting Members. https://www.hyperledger.org/about/members.

2012]. International Air Transport Association, A: Master Short-Term Engine Lease Agreement. http://www.awg.aero/assets/docs/Commentary-STEEL.PDF.

2017]. International Air Transport Association: Guidance Material and Best Practices for Aircraft Leases, 4th ed. International Air Transport Association.

2018]. International Air Transport Association: Blockchain in Aviation. https://www.iata.org/publications/Documents/blockchain-in-aviation-white-paper.pdf.

2001]. International Civil Aviation Organization, I.: The Convention on International Interests in Mobile Equipment, and Its Protocol on Matters Specific to Aircraft Equipment. https://www.unidroit.org/instruments/security-interests/cape-town-convention.

2013]. International Civil Aviation Organization: Annex 15 to the Convention on International Civil Aviation: Aeronautical Information Services. International Civil Aviation Organization.

2019]. Intel Corporation: Hyperledger Sawtooth Documentation. https://sawtooth.hyperledger.org/docs/core/releases/1.0/contents.html.

Kehoe, L., Hallahan, J., 2017. Blockchain -– a game changer in aircraft leasing? Air Finance Journal10Air Finance J. 1, 84–87 https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Tax/ie-blockchain-a-game-changer-in-aircraft-leasing.pdf.

Lao, L., Li, Z., Hou, S., Xiao, B., Guo, S., Yang, Y., 2020. survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling. ACM Computing Surveys (CSUR) 53 (1), 1–32.

Li, J., Li, N., Peng, J., Cui, H., Wu, Z., 2019. nergy consumption of cryptocurrency mining: A study of electricity consumption in mining cryptocurrencies. Energy 168, 160–168 http://www.sciencedirect.com/science/article/pii/S0360544218322503.

Meehan, J., Aslantas, C., Zdonik, S., Tatbul, N., Du, J., 2017. Data ingestion for the connected world. CIDR.

Nadimi, F., 2019]. How Sanctions are Affecting Iran's Airline Industry. https://www.washingtoninstitute.org/policy-analysis/view/how-sanctions-are-affecting-irans-airline-industry.

Narayanan, A., Clark, J., 2017. itcoin's academic pedigree. Commun. ACM6012 (Nov-Commun. ACM 60 ()), 36–45, http://dx.doi.org/10.1145/3132259.

Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S., 2017]. Performance analysis of private blockchain platforms in varying workloads. 2017 26th International Conference on Computer Communication and Networks (ICCCN), 1–6.

Portal, E.D., 2020]. Analytical Report 12: Business-to-Government Data Sharing, Tech. Rep. https://www.europeandataportal.eu/sites/default/files/analytical_report_12_business_government_data_sharing.pdf.

Radin, A., Reach, C., 2017]. Russian Views of the International Order, 1st ed. RAND Corporation.

Rubio, J.E., Alcaraz, C., Rios, R., Roman, R., Lopez, J., 2020]. Distributed detection of apts: consensus vs. clustering. European Symposium on Research in Computer Security, 174–192.

2019]. S7 Airlines: The Amount of Operations via the s7 Airlines Blockchain Platform and Alfa-Bank Exceeded $1 Million in July. https://www.s7.ru/en/about/news/the-amount-of-operations-via-the-s7-airlines-blockchain-platform-and-alfa-bank-exceeded-1-million-in-july/.

Tamayo, D., 2016]. Making Blockchain Real for Business: Explained. https://www.slideshare.net/DiegoDiaz49/1-ibm-blockchain-explained.

2017]. The Hyperledger Project: Hyperledger Architecture, Volume 1: Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf.

2018]. The Hyperledger Project: Hyperledger Blockchain Performance Metrics. https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf.

2008]. Transport Studies Group: Dynamic Cost Indexing. Innovative Cooperative Actions of R&D in EUROCONTROL Programme CARE INO III.

Vranken, H., 2017. ustainability of bitcoin and blockchains. Current Opinion in Environmental SustainabilityCurr. Opin. Environ. Sustain. 28, 1–9, sustainability governance http://www.sciencedirect.com/science/article/pii/S1877343517300015.

Yang, Y., Cooper, D., Collomosse, J., Dragan, C., Manulis, M., Briggs, J., Steane, J., Manohar, A., Moncur, W., Jones, H., 2020. Tapestry: A de-centralized service for trusted interaction online. IEEE Transactions on Services ComputingIEEE Trans. Serv. Comput.

Yermack, D., 2017. orporate governance and blockchains. Review of Finance 21 (1), 7–31.