Blockchain Model for Enhancing Aircraft Maintenance Records Security

Ahrash Aleshi
Department of Elecrical, Computer, Software, and Systems Engineering
Embry-Riddle Aeronautical University
Daytona Beach, FL, USA
aleshia@my.erau.edu

Remzi Seker
Department of Elecrical, Computer, Software, and Systems Engineering
Embry-Riddle Aeronautical University
Daytona Beach, FL, USA
sekerr@erau.edu

Radu F. Babiceanu
Department of Elecrical, Computer, Software, and Systems Engineering
Embry-Riddle Aeronautical University
Daytona Beach, FL, USA
babicear@erau.edu

***Abstract*—This paper addresses the need to enhance the integrity and transparency of aircraft maintenance records in the aviation industry by using blockchain technologies. A physical aircraft maintenance logbook is susceptible to being lost or destroyed. A nonexistent aircraft maintenance logbook hurts the confidence in integrity and reputation of the aircraft. Furthermore, fraud can occur through forgery of Federal Aviation Administration (FAA) personnel signatures and the installation of non-official aircraft parts with implications to homeland security. The scope of this work is to develop a secure blockchain that can store aircraft service records and information in a digital distributed ledger. By keeping the maintenance logbook on a digital ledger, records can be stored indefinitely in a trusted environment with the integrity of records guaranteed. Additionally, to achieve being a distributed ledger, a consensus algorithm Proof of Elapsed Time (PoET) is used to display the global state accurately to all users. The proposed Secure Aircraft Maintenance Records (SAMR) blockchain uses the Linux Foundations open sourced software "Hyperledger" to facilitate an environment that mimics a real-world implementation. The blockchain logic for SAMR implementation was made in Python through creation of a permission-based blockchain for holding the maintenance records.**

Keywords—blockchain, aircraft maintenance records, system security, PoET consensus algorithm

I. INTRODUCTION

Aircraft maintenance records are often an afterthought for many aircraft owners. However, keeping accurate records is a critical procedure in maintaining an aircraft's airworthiness. Currently, aircraft maintenance logbooks are stored in a physical ledger located on the aircraft or kept in the owner's possession. A major risk associated with maintaining a physical logbook is loss or theft, both of which will render the aircraft un-airworthy. While the aircraft may be in perfect shape to fly, it is not airworthy without the ledger showing that proper maintenance has been completed [1]. These regulations are part of the Federal Aviation Administration (FAA) CFR Part 43: Maintenance, Preventive Maintenance, Rebuilding, and Alterations [2]. If the logbook is lost, the aircraft owner will have to reconstruct it from scratch, which increases in difficulty, as the aircraft gets older. Reconstructing the maintenance history of an aircraft is extremely expensive and time consuming, and often requires additional documents to satisfy the FAA requirements. To begin the reconstruction of maintenance logs, FAA Advisory Circular 43-9C, directs referencing other records that reflect the time in service, research records maintained by repair facilities, and reference records maintained by individual mechanics [1]. As the aircraft becomes older, repair facilities and mechanics may no longer be in business or hold the recorded log of events for the aircraft any more. If certain record parts are still not able to be reconstructed, an aircraft owner will need to make a notarized statement in the new record describing the loss and establishing the time in service based on the research and the best estimate of time [1].

Additionally, at the time of sale of the aircraft, a lost maintenance logbook can affect the resale value by as much as 50% as stated by various aviation insurance companies [3]. Even if notarized statements have been made, potential customers are skeptical about buying an aircraft without the original documents. An aircraft buyer will typically look at the following items in a maintenance logbook to ensure compliance and aircraft continuity [4].

- The aircraft's total maintenance history.
- Total airframe, engine, and propeller time.
- Compliance with airworthiness directives (AD).

If the aircraft's maintenance logbooks do not demonstrate compliance or continuity, the buyer of the aircraft would need to purchase it at a lower cost to cover the cost of reconstructing the records. With a physical logbook, the falsification of maintenance records is also inherently possible through the forgery of signatures of an FAA inspection authority. While rare, the FAA takes a strict stance to ensure the integrity of aircraft maintenance records by prosecuting those responsible. Given the potential homeland security implications, the FAA is warranted to act in such a manner to avoid falsification of maintenance information for any reason, since it is pertinent to aviation safety. The FAA 14 CFR 43.12 title prohibits the falsification, reproduction, or alteration of maintenance records [5]. Any operator found in violation of the title would have their certificate suspended or revoked.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

II. Secure Aircraft Maintenance Records System

The objective of this work is to address the flaws of using a physical aircraft maintenance logbook by creating a secure and transparent blockchain ledger that will hold Aviation Maintenance Records. Blockchain technology was proposed as a disruptor in aviation operations in areas such as: (a) frequent flier points, (b) baggage, cargo, and spare parts, (c) distribution and payment, (d) passenger and crew identity management, and (e) smart contracts across the travel value chain [6]. The FAA has strict guidelines and regulations on when aircraft owners and operators should record any maintenance done on the aircraft but does not mandate the form of which they can be stored. Hence, we propose Secure Aircraft Maintenance Records (SAMR), a computationally secure blockchain to store aircraft records indefinitely, transparently, and securely.

The proposed SAMR blockchain runs off a global collaboration open source project hosted by the Linux Foundation named Hyperledger. Hyperledger, at the time of this writing, is currently working on eight different blockchain technology projects that differ in their intended uses [7]. Project Sawtooth, under the umbrella of the Hyperledger, was selected to be the development environment for the blockchain because of its ability to provide security and auditability. The blockchain developed in this work uses a distributed ledger data structure to allow all participants and authorized parties to view the same information. This approach promotes reliability, transparency, flexibility, and FAA compliance, needed characteristics for the distributed ledger, defined as follows:

- **Reliability:** Refers to the ability of the blockchain to sustain information in the event of a crash or potential attack by distributing its ledger on multiple nodes located in geographically disparate locations.
- **Transparency:** The FAA, National Transportation Safety Board (NTSB), and aircraft owners can benefit from being able to see the complete history of an aircraft with proper authorization and consensus among parties.
- **Flexibility:** The modular capabilities of the blockchain allow it to remain sustainable in the ever-evolving aviation and high technology industry.
- **Compliance:** Current FAA Requirements outlined in CFR part 43 and 91 [2, 8] are complied with, making blockchain a valid choice for replacing the physical maintenance record logbook.

To achieve these blockchain benefits, three basic components of a distributed ledger are used, namely a data model, language of transactions, and a protocol [7]. The data model used is a Radix Merkle Tree on an addressable 35-byte block used to store large amounts of records. The Python programming language is then used to change the state of the ledger under parameters set on transactions (i.e., new entries). Finally, Proof of Elapsed Time (PoET) is the chosen protocol used to provide consensus and ensure all data are presented correctly in the blockchain.

This work focuses on the physical recording of aircraft maintenance including parts, inspections, and services in the private aviation industry. The commercial industry is subject to additional regulations by the FAA, along with practices that differ between all flight operations. The blockchain will adhere to all current FAA regulations that outline retention requirements for maintenance record logbooks. In addition to the new blockchain, a simulation has been developed to highlight a real-world scenario.

With blockchain technology being used in the financial industry to record transaction data, there is an opportunity to store entire aircraft maintenance records. Moving aircraft maintenance records to a secure digital solution can help with many issues that aircraft owners face trying to preserve the physical logbook. Using the approach on how to timestamp a digital document [9], along with the proven track record of cryptocurrency such as Bitcoin [10], this work proposes a new solution for the aviation industry to continue its advancement in technology and security.

III. SAMR Blockchain Model

A. Blockchain Security

Data integrity is achieved with the use of hashing data in the blockchain. By using hashes, the blockchain can be verified to not have been tampered with or altered. The process of hashing refers to "any function that takes input of some length and compresses them into short, fixed length outputs" [11]. Each block in the blockchain computes a hash value into what is called a digest. The digest relays proof to the user that the data in the block is secure and the integrity remains valid. If unauthorized modification to the block occurred, the digest value would change and results in an error on the blockchain. Bitcoin and the SAMR blockchain use two similar methods of hashing data, but which are but different in the computation of message digest.

There are numerous types of hashing algorithms but only a few are appropriate for use in secure cryptography. The approved hash functions are required to be collision resistant, include pre-image resistance, and have a second pre-image resistance [11]. Collision resistant refers to the ability of the algorithm to create different outputs from the same inputs of value m and m'. If the outputs of m and m' are the same, an attacker could run the hash algorithm to find the value of the hashed message. Next, the pre-image resistance property ensures that the hash algorithm is a one-way function and an adversary could not reverse the hash digest. Lastly, a second pre-image resistance property is also required which makes "it computationally infeasible to find a second input that has the same hash value as any other specified input" [11].

Hashing algorithms that implement the above properties were approved by the National Institute of Standards and Technology (NIST), called the Secure Hash Standard (SHS), to be used in secure cryptography. From the SHS, Secured Hash Algorithms (SHA) were specified which include SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 [11]. While each algorithm achieves different objectives and are used in various use cases, they all achieve the same three properties outlined by NIST. For comparison, Bitcoin uses SHA-256 while the proposed SAMR blockchain uses SHA-512. The differences can be highlighted in the use cases that each platform uses the hashing. Bitcoin uses SHA-

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

256 to create the Proof of Work algorithm and to create the users' addresses while the SAMR blockchain uses SHA-512 to hash the contents in the batch to verify they have been unmodified. However, the structure of SHA-256 and SHA-512 are largely identical with differences in the computation of the message digest. Table I below shows the different properties of SHA-256 and SHA 512.

As Table I shows, SHA-512 effectively doubles in size the parameters used to create the digest. A small benefit gained from using each algorithm is the speed of processing with 32-bit and 64-bit software respectively [11].

TABLE I. HASH ALGORITHM PROPERTIES

Algorithm	Message Size	Block Size	Word Size	Digest Size
SHA-256	264	512	32	256
SHA-512	2128	1024	64	512

B. Blockchain Digital Signatures

Digital signatures are used in the blockchain by the actors to ensure newly created blocks are authorized and legitimate. By using digital signatures with the SAMR system, an extra layer of security is achieved in addition to the hashes generated by the SHA-512 algorithm. The SAMR blockchain uses an approved method to create the signatures as outlined in the Digital Signature Standard (DSS) published by the NIST [12]. A variant of the Digital Signature Algorithm (DSA) mentioned in the DSS is used called the Elliptical Curve Digital Signature Algorithm (ECDSA) [12]. This variant of DSS uses elliptical curve cryptography along with secure parameters to generate secure public and private key pairs. The algorithms used are outlined in the Standards for Efficient Cryptography, which provide elliptic curve domain parameters [13]. The parameter used is called the `secp256k1`, which creates a 512-bit key pair for use in the blockchain. ECDSA and its parameters have been proven secure if the random number generator (RNG) implementation has been followed correctly [14]. By comparison, Bitcoin uses the same digital signature scheme that has proven secure from its launch in 2008. The blockchain first digitally signs the block with the private key on the transaction header block and then verifies that the public key associated with the actor is legitimate. Signatures are then used to authenticate the identity of the actors in the blockchain during the process of submitting a transaction for validation. Validators will only submit changes to the global state of the blockchain if the signatures are valid when transactions are submitted. By using digital signatures, the blockchain doubles its security layer in combination with hashes by authenticating the transactions from actors.

C. Blockchain Access Control

Access control in blockchains vary depending on the use case and development process and it must be considered during the development phase. The SAMR blockchain presented in this work uses a permission-based network and system because it was designed to be implemented as an enterprise application. Blockchains such as Bitcoin are considered public blockchains because anyone can use their service to create and receive transactions of the cryptocurrency.

Being permission-based allows the system administrator to establish roles with actors in the SAMR blockchain. The system administrator who globally sets up the blockchain for all parties is responsible for implementing access control in the system. For the scope of this work, the system administrator is run from the same virtual machine as the blockchain network and actors resided on. The limited actors in the blockchain enable it to keep it internally secure through authentication and authorization processes. Authentication layers are used to ensure that only authorized actors are submitting transactions. The verification of transactions occurs on nodes that are installed and run in the background of a Certified Repair Station (CRS) and governing bodies' computers. The network of nodes also provides consensus using the PoET algorithm to ensure the environment is fair and globally updated. Access to download and running of a node is controlled by the network administrator and given on a case-by-case basis. To be added to the network as a node, the system administrator must do their due diligence to confirm that the repair station or government entity needs the required permissions. For example, the FAA and NTSB are added as actors because of their requirement to audit maintenance records at any given time. For the scope of this work, only a few nodes for the simulation are set up and established. Two CRS stations are authorized for submitting transactions through the local network form. They have read and write permissions, to create and view transactions. The FAA and NTSB are the government agencies chosen to perform audits on the blockchain with only read permission of the blocks. All actors except the aircraft owner or operator can be used by the consensus algorithm to execute and publish a blockchain. The aircraft owner or operator does not have a node in which to validate transactions. They use a digital signature to sign off on the CRS work. A table of the actors' permissions is presented in Table II below. The simulation section of the paper further elaborates on the scenario for the actors.

TABLE II. BLOCKCHAIN ACTOR PERMISSIONS

Actor	Number of Nodes	Read Permission	Write Permission	Validate Permission
CRS	2	Y	Y	Y
Owner/ Operator	1	N	N	Y
FAA	1	Y	N	Y
NTSB	1	Y	N	Y

D. Blockchain Threat Model

Throughout the development of the blockchain, a threat model is used to discover potential weaknesses in the code and design. The threat model chosen is STRIDE, which was developed by Microsoft in an effort for developers to start thinking about threats early in the application life cycle [15]. STRIDE is an easy to remember acronym with every letter representing a different threat category, discussed below. By using the STRIDE model to develop the blockchain, attack vectors are limited as much as possible. Countermeasures such as dual layer authentication and data integrity are implemented based on the output of the threat model.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

1) Spoofing Identity: The act of spoofing refers to an adversary using the identity of an authorized user to illegally access resources on a system where they would normally not have access to. In the blockchain, spoofing could occur during the transaction submission phase where an adversary could steal the login information of the front-end form. However, dual layer security in the form of digital signatures and hashing functions makes it improbable that an adversary could submit transactions. The digital signatures verify that the user is who they say they are through authentication, and then authorizes them to use the submit button which would be the resource.

2) Tampering with Data: An adversary in this threat category would try to alter or modify any data either when it is stored or when it is in the process of transferring. The blockchain uses hashing algorithms to secure the payload during the transfer of transaction to the blockchain. The adversary would not be able to alter the data in the payload without the system knowing that the integrity of the file has been tampered with or modified. The Hyperledger Sawtooth's Journal class would then give an error back to the submitter of the transaction with the proper response. Furthermore, the storage of the approved transaction in the blockchain cannot be altered as the original block is used to generate the hash of the future blocks that are added to the blockchain. If an adversary tried to alter a previous block, the blockchain integrity would be compromised and the system administrator could revert the global state of the blockchain to before the attack. For a block to be changed, an authorized user would have to fill out a request form and submit it to the system administrator to change the block. The change would be noted on the previous block, but the old data would remain on the record and would contain a change link to the new data.

3) Repudiation: This category involves a user or attacker who leverages the inability of a system to track invalid actions and uses them to gain some advantage in the system. No threats are seen from this category in the blockchain because each section is as self-contained as possible. Proper error and response messages are also displayed for various failures.

4) Information Disclosure: Involves the leak or access of information with an individual who is not supposed to have access to it. The permission-based system in the blockchain architecture does not allow for anyone in the public to view the blockchain. Digital signatures are verified before the blockchain can be accessed and then double checked when a block ID is pulled. An adversary with no access to the system would not be able to pass through any layers. The information stored in the blockchain is also not seen as being valuable enough for this attack to occur in the general aviation industry.

5) Denial of Service: A Denial of Service (DoS) attack refers to an adversary flooding the network to disrupt the service of users. Protocols are built into Hyperledger Sawtooth's backend platform to prevent DoS attacks. Such protocols include limiting the number of incoming transactions, which can be scaled to fit future needs, and by limiting the size of the payload to a reasonable number of bits.

6) Elevation of privilege: An adversary would want to gain access to systems without having the proper access through a compromised system. The only authority responsible for establishing access roles would be the system administrator responsible for adding actors to their roles. There is no other attack vector other than the system administrator assigning roles into the system for elevation of privilege.

IV. Blockchain Code

A. Software Environment and Architecture

An open sourced copy of Ubuntu version 16.04 was used to install the Python SDK for Hyperledger Sawtooth. Along with Ubuntu, Python version 3.5 was installed as the development language for the blockchain. The blockchain and application development process was completed using the Python programming language, as it is native to Hyperledger Sawtooth. Other programming languages such as Go, Java, JavaScript, and C++ were considered but did not have the level of stability and support as Python.

To complete the program and the different classes responsible for running the blockchain, the open source program Docker Engine and Docker Compose were also used. The Docker Compose Tool is installed to run multiple applications within one docker engine instance. An example of the multiple processes being run are the REST API and Sawtooth blockchain interacting with each other when adding a block to the blockchain. To construct the Hyperledger blockchain environment in docker, a sample `.yaml` file was modified. The modified file named `docker-compose.yaml` holds startup settings for the blockchain, which include a validator, REST API, and transactions processors.

The REST API was established on TCP port 8008, which accepts submissions from the front-end form created using Flask [16]. The transactions are then submitted to the validator, which actively listens for incoming messages on TCP port 4004. The validator will then work with the transaction processor logic to either approve or deny the transaction. The settings for the blockchain were left on default to achieve the benefits of the introductory concepts. After the docker environment was set-up and verified to be running, the code for the blockchain was developed in the aircraft's Sawtooth container.

The logic behind the blockchain resides in the Aircraft package created in Hyperledger Sawtooth. The classes consist of `AircraftPayload`, `AircraftState`, `AircraftsTransactionHandler`, and the `TransactionHandler`. The logic classes' roles and responsibilities are discussed next. In addition to the Aircraft package folder, a dedicated package was created for Flask, which enables the front-end form to be created. Flask was chosen to host the local web page because for the scope of this work there was no need for a direct web hosted framework. Flask is considered a micro framework because of its limited abilities that not require libraries [16].

The classes of this folder are `app.py` and `sawtooth_client.py`. The data is parsed using a JSON data format, which interacts with the REST API in the `sawtooth_client.py` class. The template for the web form is also stored in the `app.py` folder. Together, these two

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

classes enable the user to enter in a transaction and submit it. Due to permission requirements of the blockchain, the private key is unique for each web client and will sign the transaction when it is submitted. A UML diagram of the above classes derived to develop the blockchain is presented in Fig. 1.

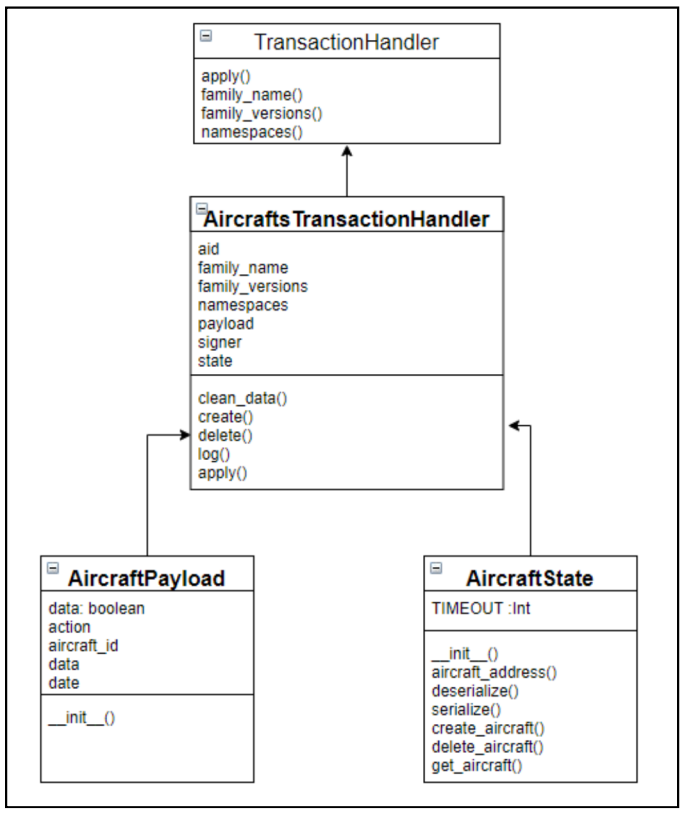

Fig. 1. Sawtooth logic UML diagram

B. *Aircraft Payload*

The aircraft payload contains information regarding the family specific attributes of the maintenance record. For example, the location of the CRS and the aircraft ID will be included in the payload to show transaction processors that it belongs to a specific transaction family. The payload is hashed using SHA-512 and verified by comparing the payload header and the payload bytes. Deserialization of the payload will occur by the transaction processor that matches the transaction family. Classes that view the payload can only see the payload bytes and not the content inside.

C. *Aircraft State*

The aircraft state is responsible for storing the hashed headers of the submitted transactions. This class is used to verify the integrity of the files in the later processes. The headers can be used to `get_aircraft` if the user possesses an `aircraft_id`.

D. *Aircraft Transaction Holder*

The main logic for the transaction processors resides in this class. It is used in conjunction with the main transaction handler class to verify transactions. The data from the front-end form is used to create transaction families. The attributes are then verified to be valid and the transaction is created. From here, the transaction is passed to the second handler class.

E. *Transaction Holder*

This class is the second part to the transaction processor, and it is responsible for connecting to the validators located on port 4004. The transaction from the class `AircraftsTransactionHandler` is pushed to the validator, which uses the journal sub-processes to add the block to the blockchain. An error will be returned if the validators fail to add the transaction.

V. Simulation Scenario

To demonstrate the blockchain working properly, the following scenario was developed to highlight its viability. Our scenario involves Charlie who is the owner of a 2015 Cessna 172S with aircraft registration number N12345. Charlie operates his aircraft out of a small general aviation airport in San Jose, CA at Reid-Hillview airport. Charlie regularly flies his aircraft and gets inspection done at the onsite repair station of a local Fixed-Based Operator (FBO) named Aero Aviation. While the FBO has only done physical logbook entries in the past, a new prototype blockchain application has begun testing at their CRS. Charlie has begun noticing that the door on his Cessna has started to vibrate midflight and wants his long time mechanic Mike at Aero Aviation to inspect the aircraft. Speaking to Mike, Charlie was interested in trying out the new blockchain to make his maintenance record logbooks more secure.

During the inspection, Mike finds that a bolt on the door needs to be replaced and needs to be ordered from a part supplier, scenario captured in Fig, 2. Knowing that the blockchain records all order history, Mike goes with a slightly more expensive, but official supplier for the Cessna 172S door bolt. Mike then enters the date, the aircraft make and model, aircraft identification code, aircraft owner name, mechanic name, authorized repair station name/location, description of maintenance, and the part order history into his front-end form.

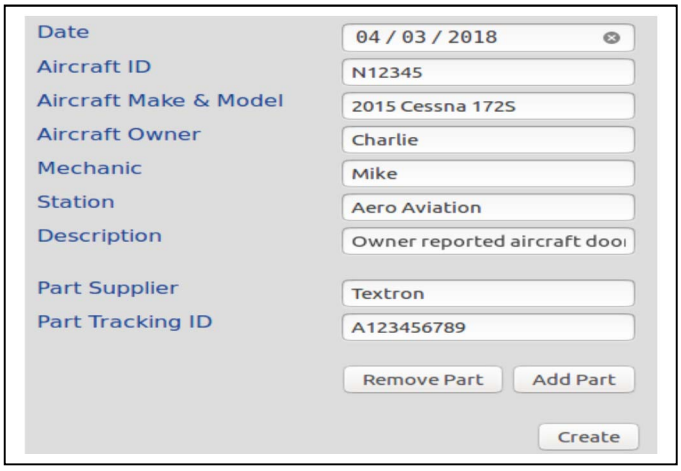

Fig. 2. Front end entry form

At this point, the repair is complete, and Charlie comes to inspect his aircraft. Mike will then click the create button which will digitally sign his mechanic signature to the transaction.

After Mike digitally signs and clicks submit, the block transaction will be pushed to the blockchain for verification using the REST API. Since all entry fields are filled out, and the integrity of the signatures has passed, the maintenance record for the repair is added to the blockchain. The events that take place in the blockchain are seen in the snapshot of Fig. 3.

Fig. 3. Successful entry of the block into the blockchain

The validators listening on port 4004 picked up a push request from the REST API. After verifying that the digital signatures are registered and approved to submit on the blockchain the block publisher sub-process in the Journal claims the block. The chain controller in the Journal class then receives the block and starts the block validation. The request is then sent to the handler where it is approved to be added to the blockchain. The chain head is then updated and the blockchain is extended. A block ID number is then presented to Mike and Charlie to easily look up and reference in the future.

The FAA regularly audits Aero Aviation to ensure they are compliant with recording aviation maintenance records. The FAA requests permission to view the block ID, which Aero Aviation authorizes. The FAA can then view the block and the information that was entered by the mechanic. With all attributes and digital signatures verified, the FAA can give approval to the FBO to continue its operations. The lookup function returns the data in the block in its serialized form. From this block, the auditors will click the link to be directed to the REST API, which will deserialize the data and show the contents of the blockchain. JSON string texts are stored on the system administrators' server, which is linked directly to the block ID. When requesting information from a block, a link is provided which redirects the user to the local host page that shows the data in a readable format (Fig. 4).

Fig. 4. Deserialized data of block contents from REST-API

VI. Conclusions

This paper proposes a secure and transparent blockchain that solves multiple issues currently plaguing the aviation industry. All issues this work set out to solve were completed along with proving it as a viable solution for implementation. Security is a high priority in the blockchain, which was why SHA-512 and digital signatures were implemented to allow for multiple checks in the transaction flow. While SHA-512 might have been unnecessary because SHA-256 has yet to be broken, it demonstrates that the blockchain is ready to be scaled for future implementations. Digital signatures that verify the integrity of the submitters of transactions were also important to close any attack vectors found when implementing the STRIDE threat model.

Transparency was achieved by using retrieved blocks from the blockchain to conduct audits. CRS and FBO can audit their own blocks before the FAA audit their maintenance record logbooks. FAA audits are made easier and are less time consuming when requesting transactions from one location. Part tracking also cuts down on any fraud occurring from the installation of non-official aircraft parts. By entering the order and/or tracking number into the blockchain, auditors can verify that the part was ordered as said by the mechanic. In addition to security and transparency, all FAA requirements for a maintenance logbook were achieved by the below implementations:

- **Record Continuity:** The blockchain achieves continuity by having a reliable network of nodes. In the rare event of a complete network crash or attack, the nodes will continue to store the latest global state offline and will be restored when brought online again
- **Includes Required Contents**: All entry fields on the digital form are taken directly from a physical maintenance record with additional fields including the part tracking and part supplier.
- **Ability to Add New Content**: Being a digital ledger gives the blockchain the ability to continue growing and storing information indefinitely. The head of the blockchain is continuously updated as blocks are added to the system and the global state is updated.
- **Provides Entry for Signatures**: Digital signatures are substituted for physical signatures to provide an extra layer of security for the blockchain, since authentication and integrity of transactions is a high priority.

By developing and verifying that all FAA requirements are met, the SAMR blockchain is a viable method for storing aircraft maintenance records. The security analysis of the proposed SAMR blockchain is the topic of future research work, with the results of it to be disseminated to interested aviation communities.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE

REFERENCES

[1] Federal Aviation Administration, "Advisory Circular: Maintenance Records," AC No: 43-9C, 2018. Available at: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_43-9C_CHG_2.pdf.

[2] Federal Aviation Administration, CFR Part 43: Maintenance, Preventive Maintenance, Rebuilding, and Alterations, 2019. Available at: https://www.govregs.com/regulations/title14_chapterI_part43.

[3] T. H. Chappell, "The unappreciated and ill-defined aircraft maintenance log," 2018. Avaiable at: http://www.aviationinsurance.com/uploads/AIRCRAFT-MAINTENANCE-LOG-BOOKS.pdf.

[4] A. Vasseur-Browne, E, "The impact of missing, lost or stolen aircraft logbooks," 2018. Avaialble at: https://www.coolinglaw.com/wp-content/uploads/2018/07/Aircraft-Logbooks-.pdf.

[5] Federal Aviation Administration, 14 CFR Part 43.12: Maintenance records: Falsifiction, reproduction, and alteration. Available at: https://www.govregs.com/regulations/title14_chapterI_part43_section43.12.

[6] International Air Transport Association, "Blockchain in aviation: Exploring the fundamentals, use cases, and industry initiatives," 2018. Available at: https://www.iata.org/publications/Documents/blockchain-in-aviation-white-paper.pdf.

[7] Sawtooth v1.0.1., Introduction, 2017. Available at: https://sawtooth.hyperledger.org/docs/core/releases/1.0.1/introduction.html.

[8] Federal Aviation Administration, CFR Part 91: General Operating and Flight Rules, 2019. Available at: https://www.govregs.com/regulations/title14_chapterI_part91.

[9] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," Journal of Cryptography, 3(2), 99-111, 1991.

[10] S. Porru, A. Pinna, M. Marchesi, R. Tonelli, "Blockchain-oriented software engineering: Challenges and new directions," Proceedings of the IEEE/ACM International Conference on Software Engineering Comapnion, Buenos Aires, Argentina, pp. 169-171, May 2017.

[11] National Institite for Standards and Technology, FIPS PUB 180-4: Secure Hash Standard, 2015. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[12] National Institite for Standards and Technology, FIPS PUB 186-4: Digital Signature Standard (DSS), 2013. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[13] Certicom Research, Standards for Efficient Cryptography 2 (SEC2), 2010. Available at: https://www.secg.org/sec2-v2.pdf.

[14] J. Katz and Y. Lindell, Introduction to Modern Cryptography, 2nd ed., Boca Raton, Francis Group, 2015.

[15] Microsoft, The STRIDE threat model, 2009. Avaialble at: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878%28v%3dcs.20%29.

[16] Flask, web development one task at a time, Version 1.0.2, 2018. Available at: http://flask.pocoo.org.

978-1-7281-5092-5/19/$31.00 ©2019 IEEE