IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Blockchain-Based Traceability and Management for Additive Manufacturing

WALA' ALKHADER[1], NOUF ALKAABI[2], KHALED SALAH[2], RAJA JAYARAMAN[1], JUNAID ARSHAD [3] and MOHAMMED OMAR[1]
[1]Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, UAE
[2]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE
[3]School of Computing and Digital Technology, Birmingham City University, Birmingham, UK

Corresponding author: Raja Jayaraman (e-mail: raja.jayaraman@ku.ac.ae)

ABSTRACT Additive Manufacturing (AM) is a major advancement in the digitization of manufacturing and production operations. Additive manufacturing uses three dimensional digital design, software and hardware equipment to precisely deposit layered materials for on-demand product manufacturing. The distinct advantages in enabling additive manufacturing includes cost efficiency, reduced time-to-market, flexibility and precise customization. However, several challenges such as trusted traceability, certification for quality compliance, and protecting intellectual property need to be addressed. Blockchain-based distributed ledgers provide tremendous advantages for product traceability and ensure trust among participating stakeholders. In this paper, we propose a blockchain-based solution for product traceability produced using additive manufacturing, guaranteeing secure and trusted traceability, accessibility, and immutability of transactions, and data provenance among supply chain stakeholders. Our proposed solution utilizes Ethereum smart contracts to govern and trace transactions initiated by participants involved in the manufacturing process. Decentralized storage of Inter-Planetary File Systems is used to store and share design files, IoT device records, and additional product specifications. We provide the system architecture, implementation, and detailed algorithms that demonstrate the working principles of our proposed solution for secure AM. Furthermore, we present detailed security and cost analysis of the solution highlighting its efficiency with respect to key security and performance requirements.

INDEX TERMS Additive Manufacturing, Blockchain, Supply Chain, 3D Printing, Cybersecurity, Trust, Traceability

I. INTRODUCTION

IN the current era of Industry 4.0, rapid advances in digitization and geographically diverse supply chains introduce an important need to track, and ensure the authenticity and origin of the products. The conventional approach employed in horizontal supply chains does not satisfy the progressions in communication systems and resulting challenges, pushing manufacturers to seek robust, trusted, flexible, and agile solutions. Additive manufacturing (AM) has provided transformational opportunities by reducing the time to deliver, enabling production facility to be closer to demand, leading to sustainable operations. AM plays an integral role in reforming the supply chain as a value-added network that incorporates all stakeholders with intertwined flows of data, products, and financial transactions amongst them. In particular, spare parts supply chain with intermittent, non-stable demand and customer dispersion can immensely benefit from the advancements in AM technology.

3D printing is synonymous to AM and has come a long way since its adoption, specifically in the area of metal 3D printing. 3D printing enables large variety of products at greater accuracy whilst minimizing material waste, which in traditional manufacturing can be as high as 90% for metal parts manufacturing. In addition, 3D printing consumes less energy and can produce parts stronger and lighter, leading to reduction in overall costs [1]. However, successful adoption of AM for products introduces several challenges related to quality and verifiable source of production, adherence to standards, and protection of copyright and intellectual property to name a few. Therefore, AM requires trusted tracking solutions and suitable technology support to be

[1]https://github.com/smartcontract9/3D-printing-using-blockchain-tech/blob/master/Remix\%20code

widely adopted across diverse industries.

Satoshi Nakamoto introduced Bitcoin [2] in 2008 using blockchain technology to create a public transaction ledger. Blockchain is a peer-to-peer, distributed, timestamped ledger of transactions managed by a cluster of computer nodes. Blockchain technology is distinguished by several fundamental characteristics, such as decentralization, persistence, anonymity, and auditability. The inherent features of blockchain technology offers significant advantages such as trustworthy, immutable, auditable transactions, eliminating intermediaries which is crucial for digital AM. Blockchain technology can provide transparent and secure data transactions, improve traceability, increase efficiency, and reduced cost of various supply chain processes.

Blockchain ledger consists of a growing chain of blocks linked and secured using cryptographic fundamentals. The blocks contain information that can represent transactions, contracts, or business rules that can be described in digital form. Every block in the chain contains the hash of the preceding block preventing any data modification leading to immutability of transactions. Blockchain is considered as a decentralized distributed ledger as it is a P2P network where the nodes of this network work on validating new blocks, those nodes maintain their copy of the chain so that the information stored in the blockchain is identical across the network. Nodes work collectively on validating and relaying transactions [3]. Blockchain technology works successfully based on three core principles: (i) cryptographic hash, (ii) digital signature (based on public and private keys), and (iii) distributed consensus mechanism (mining) [5]. Public blockchains such as Ethereum can store and execute smart contract. Smart contracts are software code that enable business logic and rules to be programmed using a high level language. For example, a manufacturer can deploy smart contract to send a secure digital spare part design or drawing and production orders to a remote manufacturing facility. Additionally, two or more entities in the supply chain can securely record an agreement over a public network without requiring a third-party authorization [4], [5].

In this paper we focus on two significant challenges in additive manufacturing using blockchain technology i.e. copyright protection of the digital product design owned by manufacturing companies, and the attestation and certification of the printed spare parts to external entities such as contract manufacturers. Leveraging blockchain technology, we highlight potential gains for additive manufacturing with respect to security and performance of decentralized 3D printing supply networks.

A. PROBLEM DEFINITION

Additive manufacturing has introduced innovative opportunities by reducing time to deliver products, enabling production facilities to be closer to demand thereby leading to sustainable operations. However, there remain challenges such as verifiable traceability of parts, trustworthy mechanisms for recording provenance, and protection of copyright and IP

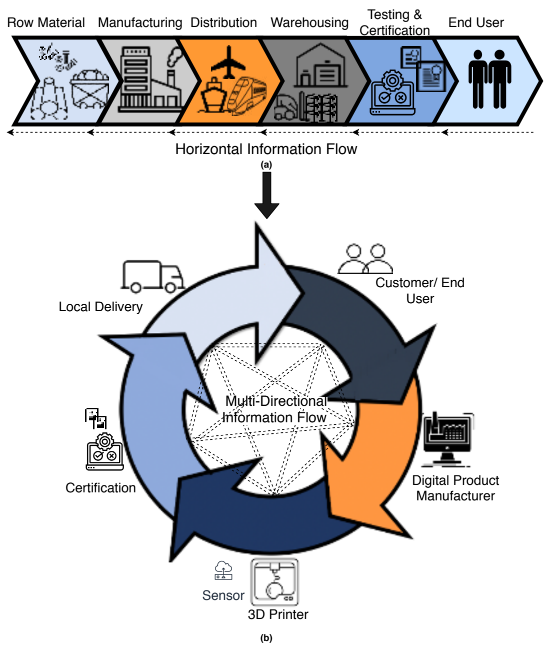


FIGURE 1: **Value-added Networks and Conventional supply chains**

for successful adoption of AM for products brings several challenges. For instance, in a complex supply chain spanning across multiple organizational and geographical domains, a trustworthy view of the product status is fundamental to achieving *just-in-time* manufacturing. Furthermore, as parts are manufactured independently, by different suppliers, adherence to the overall product specification and quality assurance becomes non-trivial. In view of these, AM requires digital innovations to improve visibility of parts across different tiers of the supply chain to achieve widespread adoption across diverse industries. Therein, this paper explores use of blockchains to address such challenges by facilitating trusted track & trace of parts across different segments of the supply chain.

B. BENEFITS OF BLOCKCHAIN FOR AM SUPPLY CHAINS

The primary advantages of adopting the proposed blockchain-based AM decentralized supply networks include:

Shorter Lead Time. Shorter lead time compared to the traditional supply chain can be achieved due to decentralized nature of the 3D printing supply network, where smaller 3D printing stations can be spatially distributed to cover larger areas demand at a considerably faster production rate.
Savings in Transportation Costs. Due to the decentralized nature of the AM network, transportation and freight costs in the traditional supply chain can be significantly minimized.
Reduced Inventory Cost. The implementation of a distributed decentralized network will shift demand from *pushed to market* into *pulled by customer demand*. So manufacturers will be following *make to order* instead of *make*

to stock, and all inventory related costs will be significantly reduced.

Product Customization. Customization of products will be flexible and doable than traditional production and supply chain methods. Resulting in greater customer satisfaction and enhanced services.

Increased Transparency and Communication. Due to the ease and openness in communication between all stakeholders in the supply chain network, all parts of the network can communicate and observe transactions. While in the traditional supply chain, only horizontal information flow is available, and communication is limited between different parties.

Reduced Carbon Footprint and Emissions. AM helps in reducing carbon footprint and emissions, causing damage to the environment. The manufacturing and supply network implementing AM and blockchain technology is substantially more environmentally friendly.

C. SCOPE AND CONTRIBUTIONS

We propose a blockchain-based solution for traceability and copyrights protection of the 3D printed digital spare parts. The main contributions are as follows:

- We discuss the advantages and use of blockchain in the additive manufacturing supply chain of spare parts.
- We highlight the opportunities that blockchain brings to additive manufacturing in spare parts supply chain, focusing on privacy and security services, traceability, copyrights, intellectual property rights, attestation, and certification.
- We develop an Ethereum blockchain-based smart contract that establishes the authenticity of the 3D digital and printed product by providing credible and secure traceability and enables attestation and certification of 3D printed products.
- We illustrate the system architecture, sequence diagrams between stakeholders, and algorithms used in Ethereum smart contracts to control and govern various interactions among stakeholders.
- We demonstrate a complete implementation of the smart contract code with testing, and present cost and security analysis.

The remainder of this paper is organized as follows. Section II presents the related work. Section III presents the proposed blockchain system architecture. Section IV describes the implementation. Section V demonstrates the functionality testing details. Section VI discusses the cost and security analysis of the implemented solution and Section VII presents the conclusions and future work.

II. RELATED WORK

In this section, we review the relevant literature under three thematic areas. We first discuss work related to additive manufacturing in spare parts management, followed by application of blockchain technology in spare parts management and additive manufacturing and finally we review literature related to Ethereum blockchain.

A. ADDITIVE MANUFACTURING IN PRODUCT MANAGEMENT

Durao et al. [7] compared the centralized factory scenario versus decentralized and distributed 3D printing-based production sites. The main focus was on work organization, network performance, and intellectual property. The results concluded that distributed manufacturing provides a highly flexible and adaptive production closer to the end-user. The authors also highlighted several challenges distributed sites might face related to information exchange, communication, security, and intellectual property protection.

Li et al. [8] presented a comparative analysis between conventional supply chain, centralized AM-based supply chain, and distributed AM-based supply chain. Their results indicate that the utilization of AM is far better than the conventional supply chain with respect to sustainable performance and environmental impact but not superior economically in all spare parts categories. The gap in economic efficiency is envisaged to be fulfilled with the advancements in technology, AM-based spare parts and its management. Khajavi et al. [9] developed a similar comparison and concluded that using AM technology, centralized production is the preferable supply chain configuration for their case. However, distributed spare parts production becomes practical as AM machines become less costly, more autonomous, and allow shorter production periods.

Sirichakwal and Conner [10] examined the role of inventory management in AM of spare parts. In particular their study investigates the impact of reduced costs of holding spare parts and lead times. Their results demonstrated that; firstly, the stock-out probability is affected by the holding costs at low demand rates. Secondly, reduced lead time could negatively impact the stock-out probability. Liu et al. [11] evaluated the impact of AM in the aircraft spare parts supply chain under three different scenarios; the conventional supply chain, centralized AM supply chain, and distributed AM supply chain. Their study concludes that the use of AM would efficiently contribute to reducing the safety inventory levels in the supply chain. Gupta et al. [20] studied the general characteristics of AM supply chain focusing on various cybersecurity risks concluding the need for robust technology to overcome supply chain security and risks.

B. BLOCKCHAIN TECHNOLOGY IN SPARE PARTS MANAGEMENT AND AM

Fernández-Caramés and Fraga-Lamas [19] presented a review on blockchain based applications for Industry 4.0 highlighting critical security challenges associated with cyber-physical systems. Mandolla et al. [4], discussed building a digital twin for AM using blockchain. The authors highlighted the unique features of blockchain that facilitate its use with the AM technology namely: transparency, traceability, and security of the blockchain that allows tracking the products through its entire history; short processing times that make transactions almost real-time, speeds up the process and allows much shorter and efficient time-to-market of the products; distributed nature of the blockchain, which helps in efficiently managing activities in the distributed AM production sites and supply chain. The authors conclude in that the implementation of blockchain in conjunction with system infrastructural elements has the potential to revolutionize and radically change the manufacturing industry.

Angrish et al. [12] proposed a prototype "FabRec" a system of a decentralized, distributed network of manufacturing stations that is fully transparent, automated, paperless, secured and verified. The authors use Ethereum smart contracts to enable decentralization and availability of the data on a peer-to-peer network. The authors provided a proof of concept system linking computing nodes, physical devices, primary CNC machines, demonstrating the feasibility of their proposed decentralized interoperable network. Furthermore, several articles have discussed the use of blockchain technology for copyright and intellectual property protection [13], [14], in which authors emphasized that regulatory requirements are a major source of impediment. However, AM is on track to provide huge numbers of parts to the market revolutionizing the way spare parts are produced, stored and handled in supply chains. Kurpjuweit et al. [21] used Delphi method to study the intergration of blockchain with AM. Their analysis provides evidence for opportunities in intellectual property (IP) digital rights management, monitoring throughout the life cycle of the printing, process improvements, and data security. Vatankhah Barenji et al. [22] presented a blockchain based platform for small and medium enterprises improving scalability, security and big-data related manufacturing problems. They validated the platform for AM application over geographically distributed supply chain stakeholders. Therefore, although the efforts discussed above have explored the use of blockchains to facilitate the emerging domain of additive manufacturing, these are limited with respect to their effectiveness to address challenges focused in this paper i.e. traceability and copyright protection of spare parts. Therein, the effort proposed in this paper takes a holistic approach, adopting an end-to-end solution which provides visibility for spare parts throughout their lifecycle whilst protecting against copyright infringement. Furthermore, leveraging smart contract technology, the proposed approach achieves end-to-end automation, minimizing human intervention which contributes towards reduced lead times for the spare parts.

C. ETHEREUM BLOCKCHAIN

Ethereum blockchain enables decentralized applications for participants to create rules, business agreements, transactions, and functions using smart contracts. Smart contracts are usually written using a high-level programming language such as *Solidity*. The growing popularity of smart contracts implemented using Ethereum blockchain have been used in diverse applications including manufacturing, supply chain, finance and insurance, healthcare and others. The versatility of smart contracts combined with cryptographic security features makes smart contracts ideal tools for transaction processing and real time data availability.

Several programming languages have been developed for writing smart contracts, and solidity is one of the most popular one for Ethereum. The author of the handbook [15] defines solidity as a JavaScript-like language developed specifically for coding Ethereum smart contracts. The solidity compiler turns the code into Ethereum virtual machine (EVM) bytecode. Smart contracts written in solidity are executed on a EVM. Smart contracts have their unique Ethereum address and can execute function calls, handle modifiers, carry arbitrary states, perform arbitrary computations, and even call other smart contracts.

The authors in [16] discussed the issue of stale blocks in the Ethereum network, which can occur when a group of mining nodes from the mining pool have more computation power than the others resulting in contributing more to the network and creating a centralization issue. A modified Greedy Heaviest Observed Subtree (GHOST) protocol is used address the issue of centralization, ensures consensus among participating nodes in the Ethereum network. Additionally, it solves the issue of stale blocks by including the stale blocks into calculations of the longest chain. To solve the centralization problem, GHOST gives 87.5 % of the block reward to the stale block, while the remaining 12.5% goes to the nephew of the stale block. By doing so, the miners will be rewarded even if their block did not become part of the blockchain.

Ethereum uses monetary units called Ether (Gwei) that can be stored into Ethereum wallets, spent, or received. Each Ethereum account contains four fields: nonce, ether balance, contract code hash, and storage root. Nonce represents the number of transactions or contracts created by an account, and it is used to ensure that each transaction can be processed only once. Ether balance is the number of Gwei in the account. Contract code hash is the keccak-256 hash of the EVM code of the account. Storage root is the 256-bit hash of the root node of a Merkle tree representing the content of the account.

Peyrott [15] indicated that the state must always be consistent across all Ethereum nodes. Although the storage is unlimited, fewer power nodes will not be able to handle it effectively. Ethereum solves this issue by using Merkle Patricia Trees, a special kind of data structure to store cryptographically authenticated data in the form of keys and values. Given the same set of keys and values, Merkle Patricia Tree can be constructed only in a single way.

A transaction is a single instruction that is cryptographically signed. Each transaction includes the recipient of the message, a signature identifying the sender, amount of Ether to be sent, an optional data field, START GAS, and GAS PRICE values. START GAS field denotes the maximum number of computational steps the transaction is allowed to consume. GAS is representing how costly is the transaction.

This limits the number of computations and solves the problem of denial of service attacks.

III. SYSTEM ARCHITECTURE

This paper is focused at two critical research challenges for additive manufacturing i.e. traceability and copyright protection of spare parts. Although there are existing efforts within application of blockchain for additive manufacturing, these are limited with respect to their feasibility to address the two challenges. **Firstly,** the proposed system takes a holistic approach, adopting an end-to-end solution which provides visibility for spare parts throughout their lifecycle whilst protecting against copyright infringement. **Secondly,** leveraging smart contract technology, the proposed approach achieves end-to-end automation, minimizing human intervention which contributes towards reduced lead times for the spare parts. **Finally** through the use of Ethereum-based smart contracts, the proposed system establishes authenticity of the 3D digital and printed product by providing credible and security traceability, enabling attestation and certification of 3D printed products.

Figure 2 presents the system architecture of our proposed solution along with its major components and interaction among them. The stakeholders in the AM supply chain includes customer, digital product manufacturer, printing workshop and Certification Authority. A customer initiates an order causing the smart contract to trigger further action in the network. The smart contract is central to our proposed solution and executes various functions, access transaction blocks, and records the history log in the blockchain ledger. The Inter Planetary File System (IPFS) is a peer-to-peer network for storing and sharing data among the stakeholders.

When a customer submits an order, the smart contract connects to the product manufacturer and the 3D printing workshop. Once the product manufacturer and the 3D printing workshop confirm accepting the order, the product designer uploads the digital design on the IPFS and hash of the file is transmitted to the 3D printing workshop. All interactions and transactions between the stakeholders are stored in the blockchain ledger. Due to storage limitations and size restrictions larger files are stored in distributed file system such as IPFS and its hash are sent to respective participants and stored in the blockchain ledger.

The 3D printing workshop will use the digital product design to print the product ordered by the customer. Throughout the printing process, cameras and IoT sensor devices will record the printing process and various environmental conditions such as temperature, vibrations, and pressure, etc. Once printing is completed, all IoT devices and camera records will be uploaded to the IPFS and hashed in the blockchain ledger. The hash for the control measures recorded during the printing process are transmitted to the Attestation and Certification Authority accessed via IPFS to verify quality control measures. Further, if the printed spare part is compliant, a notification is transmitted to the workshop and recorded on the blockchain ledger. Once the product is certified for quality compliance the product is dispatched to the customer through a local delivery, and all transactions, from workshop till delivery to customer are stored in the blockchain ledger. And finally, the smart contract ends the process once the customer confirms the delivery, and all parties reach consensus. The purpose of the consensus algorithm is to guarantee that only a single unique history of transactions exists, and history does not contain invalid or conflicting transactions.

An arbitrator is defined with technical powers to modify or reverse transactions in the system. An arbitrator is assumed to be a real-world entity which is able to perform dispute resolution. The immutable events recorded as part of the proposed system are indeed envisaged to facilitate the arbitrator, however, as the dispute resolution process is envisaged to be an offline process, this is rendered out of scope of the proposed system. During the development of the smart contract, an individual or an entity is pre-defined to be responsible for examining the produced spare part and to attest its validity, with suitable authority to ensure its implementation in compliance with the law. Smart contracts can be programmed to effectively perform the certification process. Deposits from all related stakeholders can be implemented to enforce penalties on stakeholders violating or not fulfilling their established roles in the system. In case of failure to successfully validating the process, completion ether will be transferred to the arbitrator to solve the dispute.

Secure validation of transaction is important for any transaction processing system. The asymmetric public-key cryptography employed in Ethereum blockchain helps validate transactions securely. It is based on the existence of public and private keys for each stakeholder. They are used for data encryption and data signing. The public key can be used for data encryption, where a message encrypted with the user's public key can only be decrypted by the same user private key. A user private key can be used for signing the data, which can be verified using the user's public key only. The private key cannot be derived from the public key, while the public key can be derived from the private key. In Ethereum blockchain both public and private keys are provided by the system for each user to ensure that each user has a unique identity after registering. The sequence diagram in Figure 3 explains detailed interactions between various stakeholders in the AM supply chain. Each user will be able to register and logon using their respective Ethereum address before being able to start sending or receiving transactions.

IV. IMPLEMENTATION

In this section we discuss the implementation and testing of an Ethereum smart contract for spare part AM, the smart contract is created by the manufacturer and used to track and govern the end-to-end process of ordering, designing, and printing a product till the delivery to customer. The smart contract was written in Solidity language and compiled using Remix IDE.

When a customer requests a product to be designed and printed, the manufacturer will create an object for the prod-

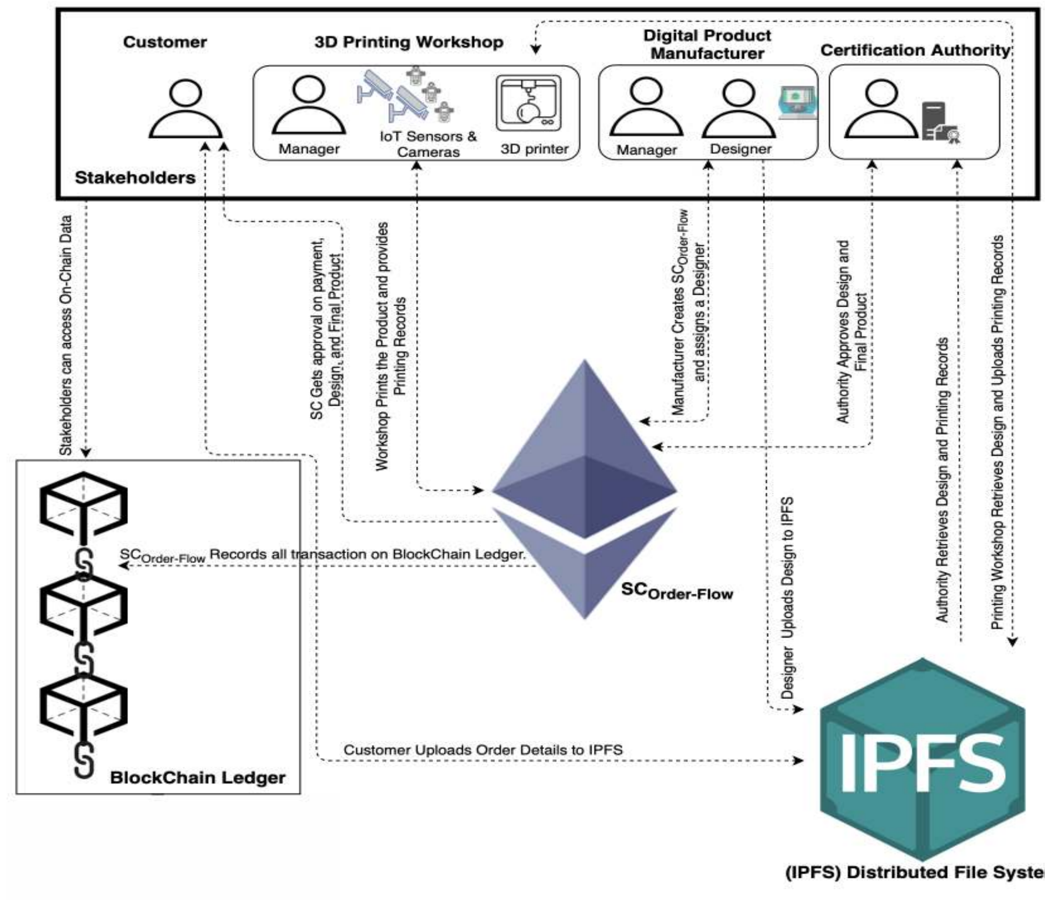

Customer
3D Printing Workshop
Digital Product Manufacturer
Certification Authority
Manager
IoT Sensors & Cameras
3D printer
Manager
Designer
Stakeholders
Stakeholders can access On-Chain Data
SC Gets approval on payment, Design, and Final Product
Workshop Prints the Product and provides Printing Records
Manufacturer Creates SC_Order-Flow and assigns a Designer
Authority Approves Design and Final Product
Authority Retrieves Design and Printing Records
Printing Workshop Retrieves Design and Uploads Printing Records
SC_Order-Flow Records all transaction on BlockChain Ledger.
SC_Order-Flow
Designer Uploads Design to IPFS
Customer Uploads Order Details to IPFS
BlockChain Ledger
IPFS
(IPFS) Distributed File System

FIGURE 2: **System architecture, major components, key stakeholders and the interactions among them**

uct, providing parameters such as a unique ID, the customer ID, and the owner ID. Initially, the owner of the smart contract will be the manufacturer. Ownership can be transferred from the product designer represented by the manufacturer, to the workshop, and finally, to the customer. Once the product object is created, the customer can upload additional details, such as the quantity and specific customization. Off-chain storage such as IPFS is used to store the digital design data and documents, only the IPFS hash will be recorded on chain and used by the smart contract and among stakeholders.

After submitting a request for spare part manufacturing, a customer receives an offer from the manufacturer with specific details on material used (metal, plastics, epoxy resin etc.), quantity, price, delivery date and specifications available. The customer, after being notified by an event in the smart contract, makes a decision to accept or reject the offer.

The customer's decision will be announced to the manufacturer. If the offer is accepted, the manufacturer will assign a designer to create and submit the product design and initiate the design phase. A request to approve the design will be sent to the customer and the Certification Authority through the function calls. When the design is approved by the Certification Authority and the customer, the 3D printing workshop receives a notification. The 3D Printing workshop will provide an offer to print the approved design, when the customer approves the workshop offer, the workshop starts printing the spare part order recording the readings of the environment using IoT devices.

After finishing the production, the workshop will request approval from the Certification Authority and the customer, similar to the design approval procedure. Once the product is approved, it will be sent to the customer through a local delivery. The customer acknowledges the receipt of the spare part and the transaction gets updated and notified to all stakeholders. Figure 3 shows the sequence diagram of the proposed distributed additive manufacturing blockchain-based supply network. We now present the detailed algorithms that represent various functions and working principles of the smart contract.

A. ADD A NEW PRODUCT

When a customer submits an order for a product, the manufacturer will initiate the function "new product" in the smart contract. Algorithm 1 details the process of adding a new object to the list of products. This request includes the manufacturer ID and the customer ID. An object of type "product" will be created with a unique ID. The owner of this product will be the manufacturer, and it will be associated with the customer through the customer ID. Customers can have several products in one order with the manufacturer,

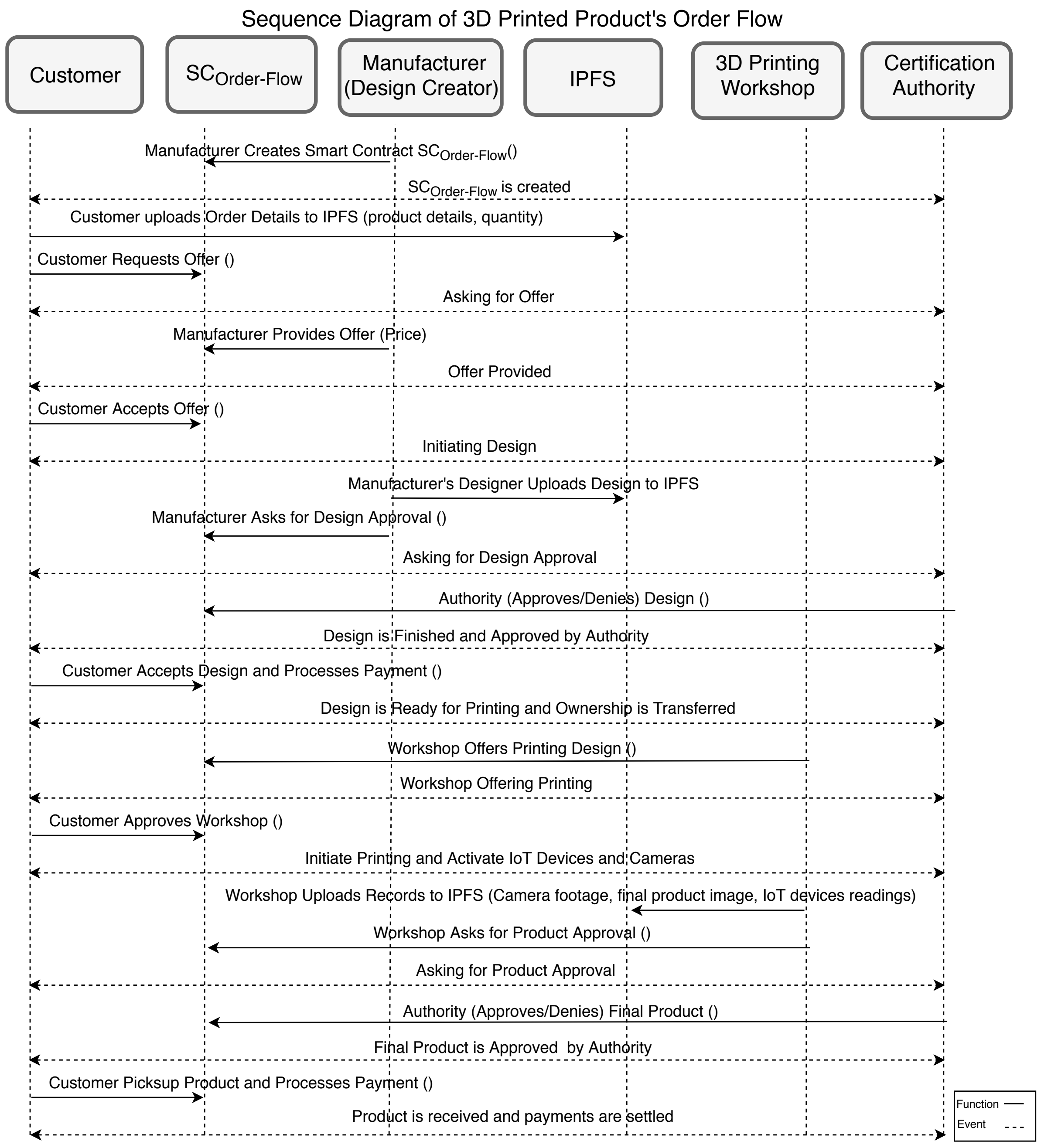

Sequence Diagram of 3D Printed Product's Order Flow
Customer
SC Order-Flow
Manufacturer (Design Creator)
IPFS
3D Printing Workshop
Certification Authority
Manufacturer Creates Smart Contract SC Order-Flow()
SC Order-Flow is created
Customer uploads Order Details to IPFS (product details, quantity)
Customer Requests Offer ()
Asking for Offer
Manufacturer Provides Offer (Price)
Offer Provided
Customer Accepts Offer ()
Initiating Design
Manufacturer's Designer Uploads Design to IPFS
Manufacturer Asks for Design Approval ()
Asking for Design Approval
Authority (Approves/Denies) Design ()
Design is Finished and Approved by Authority
Customer Accepts Design and Processes Payment ()
Design is Ready for Printing and Ownership is Transferred
Workshop Offers Printing Design ()
Workshop Offering Printing
Customer Approves Workshop ()
Initiate Printing and Activate IoT Devices and Cameras
Workshop Uploads Records to IPFS (Camera footage, final product image, IoT devices readings)
Workshop Asks for Product Approval ()
Asking for Product Approval
Authority (Approves/Denies) Final Product ()
Final Product is Approved by Authority
Customer Picksup Product and Processes Payment ()
Product is received and payments are settled
Function
Event

FIGURE 3: **Order flow sequence diagram demonstrating all the interactions between different participants of the smart contract**

and the same smart contract can track and govern all ordered products. An event will be sent to notify the customer of the creation of the new product object.

B. REQUEST AN OFFER

Algorithm 2 depicts the process of uploading the details of the product by the customer. The algorithm will take as input the customer ID, product ID, and the IPFS hash of the order details, order details can include quantity, delivery date, material, specifications, and any customization required. The smart contract will validate the authenticity of the customer and will allow the customer to provide the uploaded specifications files IPFS hash. The hash will be stored as a parameter of the product's object. The smart contract will issue an event to request the manufacturer for an offer.

C. MANUFACTURER PROVIDING OFFER

The manufacturer will provide an offer for the product requested by the customer using Algorithm 3. The manu-

Algorithm 1: Add a new product

```
   input : Manufacturer ID ,Customer ID
1  if Caller == Manufacturer then
2      Add new product ID
3      Product ID = New ID
4      Define Owner of new Product
5      Product Owner ID = Manufacturer's ID
6      Product Customer ID = Customer's ID
7      Create a notification that a new product is created
         (Event)
8  end
9  else
10     Revert contract state and show an error.
11 end
```

Algorithm 2: Request an Offer

```
   input : Customer ID,Product ID, IPFS Hash of order
           details
1  if Caller == Customer then
2      if Product Customer ID == Customer ID then
3          Product ID Specifications =
             Specifications(IPFS Hash of order details)
4          Create a notification that the customer is
             asking for offer (Event)
5      end
6      else
7          Revert contract state and show an error.
8      end
9  end
10 else
11     Revert contract state and show an error.
12 end
```

facturer ID, product ID, and the offer price will be used as inputs. Other inputs that can be considered are the delivery time, material, and quantity available. After finalizing and uploading the offer, an event will be sent by the smart contract to notify the customer with the offer and requesting customer decision, which can be either accept or reject the proposed design offer.

Algorithm 3: Manufacturer Providing Offer

```
   input : Manufacturer ID,Product ID,Price
1  if Caller == Manufacturer then
2      Product ID Price = Price.
3      Create a notification that the offer is ready.
4  end
5  else
6      Revert contract state and show an error.
7  end
```

D. RESPONSE TO DESIGN OFFER

Algorithm 4 allows the customer to accept or reject the offer provided by the manufacturer. The smart contract will check if the customer is indeed the stakeholder associated with the product. If yes, the smart contract will take the customer's decision (accept/reject) and add it to the product details. Two possible outcomes are expected, the customer may either accept the offer and then, as a result, the contract will notify the manufacturer about the customer approval and request to commence spare part designing. Otherwise, if the customer rejects an event will be issued to notify the manufacturer about the rejection.

Algorithm 4: Response to Design Offer

```
   input : Customer ID,Product ID,Decision
1  if Caller == Customer then
2      if Product Customer ID == Customer ID then
3          Customer Decision = Decision.
4          if Decision == True then
5          end
6          Create a notification that the customer
           accepted the offer and Initiate Designing
           (Event). else
7              Create a notification that the customer
                 rejected the offer (Event).
8          end
9      end
10     else
11         Revert contract state and show an error.
12     end
13 end
14 else
15     Revert contract state and show an error.
16 end
```

E. REQUEST DESIGN APPROVAL

When the design is ready, the manufacturer will upload the design files to IPFS and generate a hash, subsequently the smart contract issues an event requesting approval by the Certification Authority. This request will include the customer ID, product ID, and the IPFS hash of the uploaded spare part design to IPFS to enable the Certification Authority to examine if it meets the standards and customer design specifications. This function is presented in Algorithm (5).

F. AUTHORITY DESIGN APPROVAL

Given the product ID, the Certification Authority will be able to access the details of the product including the IPFS hash of the spare part design and any relevant documents. In Algorithm (6) we present the design approval process by the Authority. If the Authority approves the design then a notification will be issued to inform customer. However, if

Algorithm 5: Request Design Approval

```
input : Manufacturer ID,Product ID, IPFS Hash of
        Design details
1  if Caller == Manufacturer then
2      if Product Owner ID == Manufacturer ID then
3          Design ID Specifications =
             Specifications(IPFS Hash).
4          Create a notification that the Manufacturer is
             asking for Design Approval (Event).
5      end
6      else
7          Revert contract state and show an error.
8      end
9  end
10 else
11     Revert contract state and show an error.
12 end
```

the Authority rejects an event will be issued to notify the manufacturer and the customer with the rejection.

Algorithm 6: Authority Design Approval

```
input : Authority ID,Product ID,Authority-Decision
1  if Caller == Authority then
2      Authority Decision = Authority-Decision.
3      if Decision == True then

4      end
5      Create a notification that the Authority accepted
         the Design(Event).
6      else
7          Create a notification that the Authority
             rejected the Design (Event).
8      end
9  end
10 else
11     Revert contract state and show an error.
12 end
```

G. ACCEPT DESIGN, PAYMENT PROCESS AND TRANSFER OF OWNERSHIP

Algorithm (7) presents the decision and choices of the Certification Authority and customer regarding the manufacturer proposed design. If the customer accepts the design and processes the payment, the ownership of the design is transferred to the customer. The manufacturer will be notified about the decision of the customer, so interested printing workshops can offer to print and deliver the spare part. As an alternative, when the customer rejects the design, the manufacturer will be notified. Any disputes arising on can be solved by the arbitrator.

Algorithm 7: Accept Design, Payment Process and Transfer of Ownership

```
input : Customer ID,Product ID,Design Decision
1  if Caller == Customer then
2      if Product Customer ID == Customer ID then
3          Customer Design Decision = Design Decision.
4          if Design Decision == True then

5          end
6          Product Owner ID = Customer ID
7          Create a notification that the customer
             Approved the Design, payment settled and
             Ownership is transferred (Event).
8          else
9              Create a notification that the customer
                 rejected the Design (Event).
10         end
11     end
12     else
13         Revert contract state and show an error.
14     end
15 end
16 else
17     Revert contract state and show an error.
18 end
```

H. WORKSHOP OFFERING PRINTING

Once design is approved and ownership is transferred to customer. Printing workshop will receive the notification and consequently will submit an offer, and the customer will be notified through an event triggered by the smart contract as per Algorithm (8). Inputs will be Workshop ID, Product ID and the offered printing price. Other inputs that can be added are the location of the workshop, time to print the product and printer specifications.

Algorithm 8: Workshop Offering Printing

```
input : Workshop ID,Product ID,Printing Price
1  if Caller == Workshop then
2      Product ID printing Price = Printing Price.
3      Create a notification that the workshop's offer is
         ready.
4  end
5  else
6      Revert contract state and show an error.
7  end
```

I. CUSTOMER RESPONSE TO WORKSHOP OFFER

Algorithm (9) allows the customer to accept or deny the offer provided by the Workshop. The smart contract will validate if the right customer is the one associated with the product by verifying the customer Ethereum address, and if the customer approves, the smart contract will take the customer's decision and add it to he product details. Smart contract will send an event to notify the Workshop to begin Printing and Initiate recording of the quality parameters and readings of the environment using IoT devices and cameras, those readings will be needed to verify the conditions of printing and authenticate the final printed product quality and finishing.

Algorithm 9: Customer Response to Workshop Offer

```
   input : Customer ID,Product ID,Printing Decision
1  if Caller == Customer then
2      if Product Customer ID == Customer ID then
3          Customer Printing Decision = Printing
             Decision.
4          if Printing Decision == True then
5          end
6          Create a notification that the customer
             accepted the Workshop's offer,Start Printing
             and Initiate Cameras and IoT devices
             (Event).
7          else
8              Create a notification that the customer
                 rejected the Workshop's offer (Event).
9          end
10     end
11     else
12         Revert contract state and show an error.
13     end
14 end
15 else
16     Revert contract state and show an error.
17 end
```

J. REQUESTING PRODUCT APPROVAL BY CERTIFICATION AUTHORITY

Once Printing is done, the 3D prinitng Workshop will upload the final printed product details, images and recordings obtained during the printing process from cameras and IoT devices to IPFS. The quality parameters can be temperature, pressure and vibration recordings throughout the printing process, deviations from normal conditions can be reported and recorded on chain to facilitate data storage to make procedures more flexible. The 3D printing workshop adds the IPFS hash to the product details, which will trigger the smart contract to notify the Certification Authority requesting approval. This request will include the customer ID, product ID, and the IPFS hash of printing records is described in Algorithm (10).

Algorithm 10: Requesting Product Approval by Certification Authority

```
   input : Wrokshop ID,Product ID, IPFS Hash of
           Printing Records
1  if Caller == Workshop then
2      Design ID Specifications = Specifications(IPFS
         Hash of Printing Records).
3      Create a notification that the Workshop is asking
         for Product Approval (Event).
4  end
5  else
6      Revert contract state and show an error.
7  end
```

K. PRODUCT APPROVAL BY CERTIFICATION AUTHORITY

Similar to Approval/rejection of product in Algorithm (6), the decision reported by the Certification Authority and the workshop is notified with the Authority decision, the decision will be reported as a (Yes/No) value. The customer is also notified of the Authority decision. Once the final product is approved, the customer will be able to approve/reject the 3D printing workshop product is presented in the next algorithm.

Algorithm 11: Product Approval by Certification Authority

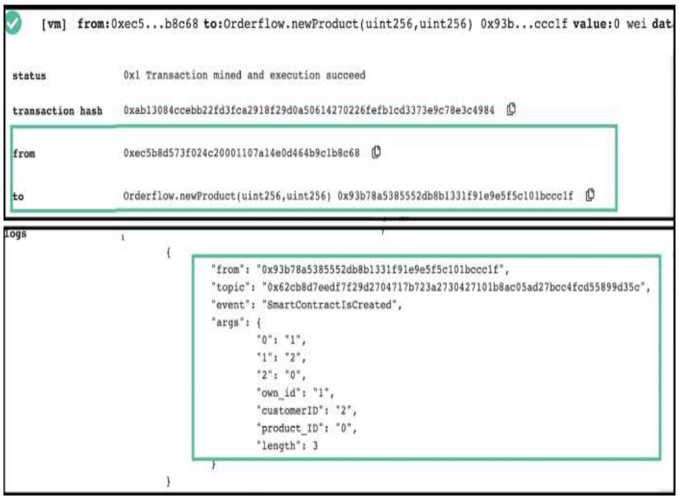
```
   input : Authority ID,Product
           ID,Authority-Final-Decision
1  if Caller == Authority then
2      Authority Decision = Authority-Final-Decision.
3      if Decision == True then
4      end
5      Create a notification that the Authority accepted
         the Product(Event).
6      else
7          Create a notification that the Authority
             rejected the Product (Event).
8      end
9  end
10 else
11     Revert contract state and show an error.
12 end
```

L. CUSTOMER PRODUCT APPROVAL AND PAYMENT

When the Certification Authority's decision is to accept the product, the customer would have the choice to either accept or reject the design, explained in algorithm 12. If the customer accepts the product and processes the payment the 3D printing workshop will be notified about the decision of the customer, the spare part will be delivered to the customer and the order will be declared as closed.

Algorithm 12: Customer Product Approval and Payment

```
input : Customer ID,Product ID,Design Decision
1  if Caller == Customer then
2      if Product Customer ID == Customer ID then
3          Customer Design Decision = Design Decision.
4          if Design Decision == True then
5          end
6          Create a notification that the customer
             Approved the Design, payment settled and
             Product is Picked-up by customer (Event).
7          else
8              Create a notification that the customer
                 rejected the Product (Event).
9          end
10     end
11     else
12         Revert contract state and show an error.
13     end
14 end
15 else
16     Revert contract state and show an error.
17 end
```



```
[vm] from:0xec5...b8c68 to:Orderflow.newProduct(uint256,uint256) 0x93b...ccc1f value:0 wei dat

status            0x1 Transaction mined and execution succeed
transaction hash  0xab13084ccebb22fd3fca2918f29d0a50614270226fefb1cd3373e9c78e3c4984
from              0xec5b8d573f024c20001107a14e0d464b9c1b8c68
to                Orderflow.newProduct(uint256,uint256) 0x93b78a5385552db8b1331f91e9e5f5c101bccc1f
logs              [
                      {
                          "from": "0x93b78a5385552db8b1331f91e9e5f5c101bccc1f",
                          "topic": "0x62cb8d7eedf7f29d2704717b723a2730427101b8ac05ad27bcc4fcd55899d35c",
                          "event": "SmartContractIsCreated",
                          "args": {
                                  "0": "1",
                                  "1": "2",
                                  "2": "0",
                                  "own_id": "1",
                                  "customerID": "2",
                                  "product_ID": "0",
                                  "length": 3
                          }
                      }
                  ]
```

FIGURE 4: **Output showing manufacturer successfully added a new product to the smart contract and alert notifying to all participants**

V. FUNCTIONALITY TESTING

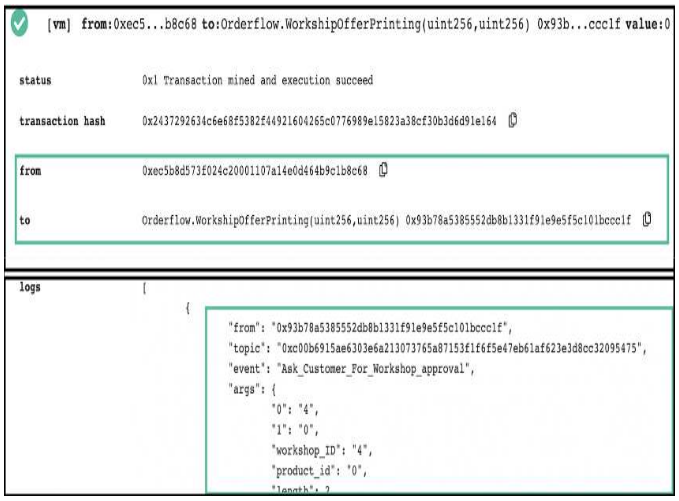
The developed smart contract was tested using Remix IDE, a versatile in-browser development and testing environment for the smart contract functions. In this section we present the results of testing, function calls showing the corresponding outputs and logs. For our testing scenarios, we assumed four participants of the AM supply chain network interacting with the smart contract, the manufacturer, customer, Certification Authority and 3D printing workshop. The Ethereum addresses of the Manufacturer, Customer, Certification Authority, 3D Printing Workshop and the Smart Contract are provided in table(1).

TABLE 1: **Ethereum addresses of all participants in the tested smart contract**

User	Ethereum Address
Manufacturer	0xAfd8741232Af159704385d54A683D3f6F50B3BB7
Customer	0x104fb6298a35E2c471E5CD8455D4E769a9121803
Authority	0xEc56b12C8DE3799d37586b512490326Fa898a5E2
Workshop	0xEc5B8D573F024C20001107a14E0D464b9c1b8C68
Contract	0x93b78a5385552db8b1331f91e9e5f5c101bccc1f

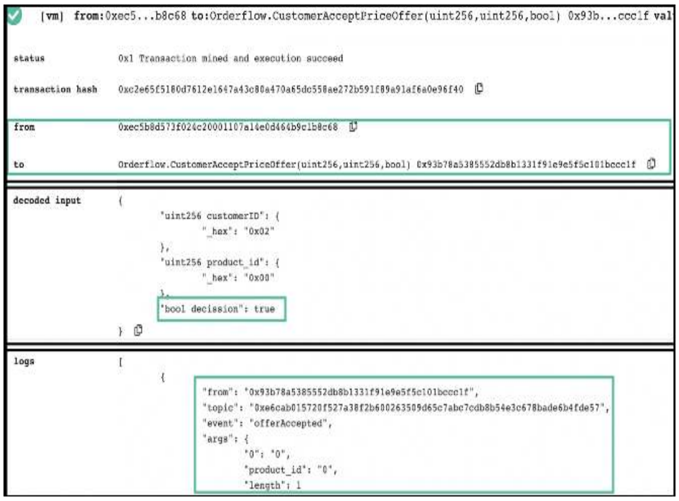
All functions have a state requirement or a condition that has been tested successfully. Several functions are similar in execution and results. Therefore, only results from testing important functions is presented here, demonstrating inputs and outputs logs resulting from the stakeholder interactions with the smart contract functions as follows:

1) Adding a new product: A new product is added by the manufacturer as shown in the logs in figure (4), the new product ID is (0), manufacturer ID is (1), customer ID is (2) and the transaction is from manufacturer EA to the Smart Contract address. These logs shows the successful addition of a new product and relating the new product to a specific manufacturer (the smart contract owner) and a specific customer, then the logs highlight that an event is issued successfully announcing the creation of the smart contract and the addition of a new product.
2) Workshop offers to printing the spare part: Both the manufacturer transferring design of the product and the offer from workshop to print the spare part according to the design provided are similar in execution. We present the scenario where the workshop offer is shown in figure (5) as an example of the successful implementation, where transaction workshop ID is (4) is related to the same product (ID =0), and the logs shows that an event is issued to request customer approval based on the workshop offer.
3) The scenario where customer accepts the design offer is similar to customer accepting workshop offer are shown in figure (6). The input shows customer decision (true) and the logs show a successful event by the smart contract that the offer related to product (ID=0) was accepted.
4) Request price approval from customer: Figure (7) illustrates the scenario from manufacturer to the smart contract, the inputs are offered price (Price =5) and the transaction logs demonstrate the event triggered by the smart contract asking for acceptance of the manufacturer's offer related to product (ID=0) and customer (ID=2).



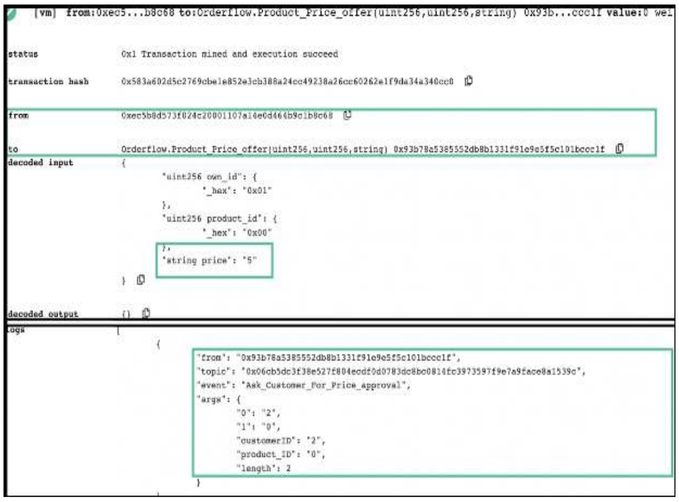
```
[vm] from:0xec5...b8c68 to:Orderflow.WorkshipOfferPrinting(uint256,uint256) 0x93b...cccl f value:0

status              0x1 Transaction mined and execution succeed

transaction hash    0x2437292634c6e68f5382f44921604265c0776989e15823a38cf30b3d6d91e164

from                0xec5b8d573f024c20001107a14e0d464b9c1b8c68

to                  Orderflow.WorkshipOfferPrinting(uint256,uint256) 0x93b78a5385552db8b1331f91e9e5f5c101bccc1f

logs                [
                        {
                            "from": "0x93b78a5385552db8b1331f91e9e5f5c101bccc1f",
                            "topic": "0xc00b6915ae6303e6a213073765a87153f1f6f5e47eb61af623e3d8cc32095475",
                            "event": "Ask_Customer_For_Workshop_approval",
                            "args": {
                                "0": "4",
                                "1": "0",
                                "workshop_ID": "4",
                                "product_id": "0",
                                "length": 2
```

FIGURE 5: **Logs showing workshop (ID=4) requesting approval for its offer to print product (ID=0).**



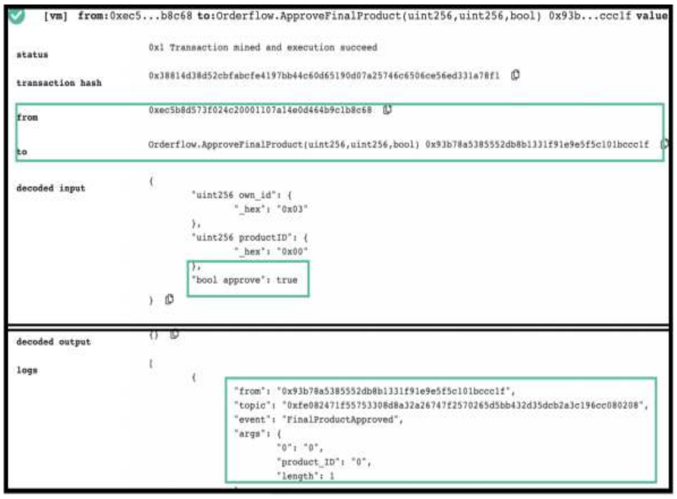
```
[vm] from:0xec5...b8c68 to:Orderflow.CustomerAcceptPriceOffer(uint256,uint256,bool) 0x93b...ccc1f val

status              0x1 Transaction mined and execution succeed

transaction hash    0xc2e65f5180d7612e1647a43c80a470a65dc558ae272b591f89a91af6a0e96f40

from                0xec5b8d573f024c20001107a14e0d464b9c1b8c68

to                  Orderflow.CustomerAcceptPriceOffer(uint256,uint256,bool) 0x93b78a5385552db8b1331f91e9e5f5c101bccc1f

decoded input       {
                        "uint256 customerID": {
                                "_hex": "0x02"
                        },
                        "uint256 product_id": {
                                "_hex": "0x00"
                        },
                        "bool decission": true
                    }

logs                [
                        {
                            "from": "0x93b78a5385552db8b1331f91e9e5f5c101bccc1f",
                            "topic": "0xe6cab015720f527a38f2b600263509d65c7abc7cdb8b54e3c678bade6b4fde57",
                            "event": "offerAccepted",
                            "args": {
                                    "0": "0",
                                    "product_id": "0",
                                    "length": 1
```

FIGURE 6: **Execution of a successful acceptance of design offer by the customer**



```
[vm] from:0xec5...b8c68 to:Orderflow.Product_Price_offer(uint256,uint256,string) 0x93b...ccc1f value:0 wei

status              0x1 Transaction mined and execution succeed

transaction hash    0x583a602d5c2769cbe1e852e3cb388a24cc49238a26cc60262e1f9da34a340cc0

from                0xec5b8d573f024c20001107a14e0d464b9c1b8c68

to                  Orderflow.Product_Price_offer(uint256,uint256,string) 0x93b78a5385552db8b1331f91e9e5f5c101bccc1f
decoded input       {
                        "uint256 own_id": {
                                "_hex": "0x01"
                        },
                        "uint256 product_id": {
                                "_hex": "0x00"
                        },
                        "string price": "5"
                    }

decoded output      {}
logs                [
                        {
                            "from": "0x93b78a5385552db8b1331f91e9e5f5c101bccc1f",
                            "topic": "0x06cb5dc3f38e527f804ecdf0d0783dc8bc0814fc3973597f9e7a9face8a1539c",
                            "event": "Ask_Customer_For_Price_approval",
                            "args": {
                                    "0": "2",
                                    "1": "0",
                                    "customerID": "2",
                                    "product_ID": "0",
                                    "length": 2
                            }
```

FIGURE 7: **Logs showing successful provision of an offer to the smart contract by the Manufacturer and the announcement to the customer**



```
[vm] from:0xec5...b8c68 to:Orderflow.ApproveFinalProduct(uint256,uint256,bool) 0x93b...ccc1f value

status              0x1 Transaction mined and execution succeed

transaction hash    0x38814d38d52cbfabcfe4197bb44c60d65190d07a25746c6506ce56ed331a78f1

from                0xec5b8d573f024c20001107a14e0d464b9c1b8c68

to                  Orderflow.ApproveFinalProduct(uint256,uint256,bool) 0x93b78a5385552db8b1331f91e9e5f5c101bccc1f

decoded input       {
                        "uint256 own_id": {
                                "_hex": "0x03"
                        },
                        "uint256 productID": {
                                "_hex": "0x00"
                        },
                        "bool approve": true
                    }

decoded output      {}

logs                [
                        {
                            "from": "0x93b78a5385552db8b1331f91e9e5f5c101bccc1f",
                            "topic": "0xfe082471f55753308d8a32a26747f2570265d5bb432d35dcb2a3c196cc080208",
                            "event": "FinalProductApproved",
                            "args": {
                                    "0": "0",
                                    "product_ID": "0",
                                    "length": 1
```

FIGURE 8: **A successful approval granted to the smart contract of the final product.**

5) Certification Authority Approves the final product, a transaction from Authority (ID=3) to the smart contract approving the final product (ID=0) as shown in the inputs where the decision is (true). Therefore, an event is triggered to announce approval by the Authority of the final printed product to be received by the customer. figure (8) presents the successful transaction logs.

VI. DISCUSSION AND ANALYSIS

In this section we will present the cost analysis for the Ethereum smart contract, followed by security analysis, and finally some generalization and extensions on this paper's work are discussed.

A. COST ANALYSIS

As our implementation and execution are carried using the Ethereum blockchain, every transaction processed on the blockchain network consumes or is paid for with gas, which is effectively paid in *Ether* (Ethereum currency).

The total cost of each function performed on the blockchain network consists of two parts the transaction and execution gas costs. The execution cost represents the cost of the actual execution of the function code handling the translation on the blockchain network. It includes the cost of the internal storage in the smart contract as well as any manipulation with the state. Moreover, the transaction cost includes other factors related to the deployment of the contract and sending the data to the blockchain network [18].

Table 2 shows the gas costs of the functions in the smart contract as well as their prices in US Dollars. The gas price used in Table 2 shows fastest (10.5 Gwei), the average gas (6.9 Gwei), and the Slow (5 Gwei) price on April 10th, 2020, according to the ETH Gas Station [17]. The functions in Table 2 are executed by the Manufacturer, Customer, Work-

TABLE 2: **Gas costs of the smart contract functions.**

Function	Function Caller	Transaction Cost	Execution Cost	Gas used	Fastest Cost$	Average Cost$	Slow Cost$
Create SC	Manufacturer	3229184	2397780	5626964	$9.21696	$6.05687	$4.38903
Manufacturer Adds new product	Manufacturer	86673	65017	151690	$0.0194	$0.16329	$0.11833
Customer Asking for Offer	Customer	28016	6104	34120	$0.05589	$0.03672	$0.02661
Manufacturer Providing Offer	Manufacturer	54202	32034	86236	$0.14126	$0.09282	$0.06727
Customer Accepts Price Offer	Customer	26107	4323	30430	$0.04984	$0.03276	$0.02374
Manufacturer Asks for Design Approval	Manufacturer	48419	26315	74734	$0.12241	$0.08045	$0.0583
Authority Approves design	Authority	46259	24475	70734	$0.11586	$0.07614	$0.05518
Customer Approves Design	Customer	31340	9556	40896	$0.06699	$0.04402	$0.0319
Workshop Offers Printing	Workshop	25356	3764	29120	$0.0477	$0.03134	$0.02271
Customer Approves Workshop	Customer	68787	47003	115790	$0.18966	$0.12464	$0.09032
Workshop Asks for Product Approval	Workshop	25093	3501	28594	$0.04683	$0.03078	$0.02231
Authority Approve Final Product	Authority	27087	5303	32390	$0.05306	$0.03487	$0.02527
Customer Approves Final Product	Customer	25093	3501	28594	$0.04683	$0.03078	$0.02231

shop, or Authority, as seen in the Function Caller column of the table.

As shown in the table overall, the typical cost for executing individual functions are very small. This is because most changes in the state of variables conducted by the functions are relatively minor, and the costs in our smart contract operations are proportional to the changes in the state of the smart contract. Furthermore, the creation of the smart contract cost is comparatively higher than that of individual functions as every line of code within a smart contract requires certain amount of gas to be executed which therefore results in a higher cost for the smart contract creation. However, this higher cost is rendered acceptable, taking into consideration that a manufacturer can use one smart contract to operate several design orders by adding new products to the same contract.

B. SECURITY ANALYSIS

In a geographically diverse setting such as the additive manufacturing supply chains, secure operation underpins successful adoption of technological interventions. In particular, integrity, accountability, authorization and non-repudiation are fundamental security properties which should be addressed by a solution to achieve trusted supply chains within additive manufacturing. In this section, we present an overview of how our proposed solution addresses these security requirements with empirical evaluation envisaged to be conducted as part of our future work.

Integrity. For each product, the transaction history and data must be available for the users to track and trace the product to its origin. Our blockchain solution ensures the integrity of all the events and logs leveraging cryptographic fundamentals of the blockchain technology. Moreover, product design, readings, and images are stored on the IPFS distributed servers and only storing the hash on the chain. This ensures efficient utilization of on-chain storage thereby facilitating scalable and performance efficient operation.

Accountability. Efficient auditing is paramount to achieving accountability within supply chains. Within our proposed solution, all events are logged in the tamper-proof blockchain ledger thereby providing a secure log of all transactions occurring within the AM supply chain. Furthermore, each participating entity is allocated a unique Ethereum address which facilitates identification of entities within the supply chain. Through tamper-proof log of events and Ethereum-based identification of stakeholders, each participating entity is accountable for its actions in the blockchain since each caller is traced back to their Ethereum address.

Non repudiation. Leveraging cryptographic fundamentals of the blockchain technology, all transactions within the AM supply chain are stored within a tamper-proof log of events associated within unique Ethereum addresses for each entity. Consequently, an entity cannot deny their actions on the blockchain since all of the transactions are cryptographically signed and saved in the tamper-proof logs.

Authorization. Each participating entity is authorized to execute only specific functions in the smart contract. This is achieved by verifying the Ethereum address of the caller before allowing him to execute a function.

MITM and Replay Attacks. The security features of the blockchain facilitates protection against a MITM attack since each participant's private key cryptographically signs every transaction initiated by the participant. Therefore, an intruder will not be able to change the transaction content without having the private key. Moreover, duplicate transactions generated by replay attacks will be discarded by the mining nodes.

VII. CONCLUSION

In this paper, we have provided a blockchain-based solution for proof of authenticity of a 3D printed product through the secured traceability of the printing process. Attestation and certifications of a 3D printed product form a significant challenge, and this paper demonstrates that the use of blockchain, Ethereum smart contract, and IPFS can enable trusted and authenticated traceability throughout the process. The proposed work has also demonstrated that 3D printed products combined with the fundamental security and immutability properties of blockchain can be authenticated and approved at a much faster time and at a minimal cost. Specifically, the cost estimate is always under 0.15 USD per transaction which indicates that deploying such a solution can be cost efficient.

The proposed work can be extended to include bidding of different manufacturers and different printing workshops so that a customer will have several choices among which they can make a choice. An interesting avenue of research within this direction will be a reputation system to enable categorization of bidders. Such a system will be vital for trustworthy manufacturing service and is envisaged to use historical data such as contracts, number of orders, number of disputes, and other factors to calculate a reputation for each entity.

ACKNOWLEDGMENT

This publication is based upon work supported by the Khalifa University of Science and Technology under Award No. RCII-2019-002 - Center for Digital Supply Chain and Operations Management.

REFERENCES

[1] L. E. Murr, "Frontiers of 3D printing/additive manufacturing: from human organs to aircraft fabrication", Journal of Materials Science & Technology, vol. 32, no. 10, pp. 987-995, 2016.

[2] M. Swan, "Blockchain: Blueprint for a new economy" ,O'Reilly Media, Inc.,2015.

[3] T. Prathyusha, M. Kavya, and PSL. Akshita, "Blockchain technology", International Journal of Computer & Mathematical Sciences, vol. 7, no. 3, pp. 232-237,2018.

[4] C. Mandolla, A. M. Petruzzelli, G. Percoco,and A. Urbinati, "Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry", Computers in Industry, vol.109, pp.134-152, 2019.

[5] J. Al-Jaroodi, and N. Mohamed, "Blockchain in industries: A survey" ,IEEE Access, vol.7, pp.6500-36515, 2019.

[6] Z. Zheng, S. Xie, H.N. Dai, X. Chen,and H. Wang, "Blockchain challenges and opportunities: A survey",International Journal of Web and Grid Services, vol.14, no.4, pp. 352-375,2018.

[7] L. F. Durão, A.Christ, E.Zancul, R.Anderl, and K. Schützer, "Additive manufacturing scenarios for distributed production of spare parts",The International Journal of Advanced Manufacturing Technology, vol. 93, no. 1-4, pp. 869-880, 2017.

[8] Y.Li, G. Jia, Y. Cheng, and Y. Hu, "Additive manufacturing technology in spare parts supply chain: a comparative study", International Journal of Production Research, vol. 55, no. 5, pp. 1498-1515, 2017.

[9] S.H.Z. Khajavi, J.Partanen, and J.Holmström, "Additive manufacturing in the spare parts supply chain", Computers in Industry, vol. 65, no. 1, pp. 50-63, 2014.

[10] I. Sirichakwal, and B. Conner, "Implications of additive manufacturing for spare parts inventory", 3D printing and Additive Manufacturing, vol. 3, no. 1, pp. 56-63, 2016.

[11] P.Liu, S. H.Huang, A.Mokasdar, H.Zhou, and L.Hou,"The impact of additive manufacturing in the aircraft spare parts supply chain: supply chain operation reference (SCOR) model based analysis", Production Planning & Control, vol. 25, no. 13-14, pp. 1169-11813, 2014.

[12] A.Angrish, B.Craver, M.Hasan, and B.Starly, "A case study for blockchain in manufacturing:"FabRec": a prototype for peer-to-peer network of manufacturing nodes", Procedia Manufacturingy, vol. 26,pp. 1180-1192, 2018.

[13] M.Holland, C.Nigischer, J.Stjepandic,"Copyright protection in additive manufacturing with blockchain approach", Transdisciplinary Engineering: A Paradigm Shift, vol. 5, pp. 914-921, 2017.

[14] A. Savelyev, "Copyright in the blockchain era: Promises and challenges", Computer law & security review, vol. 34,no. 3, pp. 550-561, 2018.

[15] S. Peyrott, "An Introduction to Ethereum and Smart Contracts",[Online]. https://auth0.com/blog/an-introduction-to-ethereum-and-smart-contracts/. Accessed on: March 19, 2020.

[16] D. Vujičić, D. Jagodić, and S. Ranđić,"Blockchain technology, bitcoin, and Ethereum: A brief overview", International symposium infoteh-jahorina (infoteh), IEEE,book, pp. 1-7, 2018.

[17] "ETH Gas Station", [Online]. Available:https://ethgasstation.info. Accessed on: April 10, 2020.

[18] H. R. Hasan, and K. Salah,"Combating deepfake videos using blockchain and smart contracts", IEEE Access, vol. 7, pp. 41596-41606, 2019.

[19] T.M. Fernández-Caramés, and P. Fraga-Lamas,"A review on the application of blockchain to the next generation of cybersecure industry 4.0 smart factories", IEEE Access, vol. 7, pp.45201-45218, 2019.

[20] N. Gupta, A.Tiwari, S.T. Bukkapatnam, and R.Karri,"Additive manufacturing cyber-physical system: Supply chain cybersecurity and risks", IEEE Access, vol. 8, pp. 47322-47333, 2020.

[21] S.Kurpjuweit, C.G.Schmidt, M. Klöckner, and S.M. Wagner, "Blockchain in additive manufacturing and its impact on supply chains", Journal of Business Logistics, 2019.

[22] A. Vatankhah Barenji, Z. Li, W.M. Wang, G. Q. Huang, and D.A. Guerra-Zubiaga,"Blockchain-based ubiquitous manufacturing: a secure and reliable cyber-physical system", International Journal of Production Research, 58(7), 2200-2221, 2020.