Confidential Token-Based License Management

Felix Engelmann
fe-research@nlogn.org
Department of Computer Science
Aarhus University
Aarhus, Denmark

Jan Philip Speichert
Ralf God
jan@idi-systems.com
ralf.god@tuhh.de
Institute of Aircraft Cabin Systems
TUHH - Hamburg University of
Technology
Hamburg, Germany

Frank Kargl
Christoph Bösch
frank.kargl@uni-ulm.de
christoph.boesch@uni-ulm.de
Institute of Distributed Systems
Ulm University
Ulm, Germany

Figure 1: Installed belt mold spare part for the Airbus A300/A310 additively manufactured on demand.

ABSTRACT

In a global economy with many competitive participants, licensing and tracking of 3D printed parts is desirable if not mandatory for many use-cases. We investigate a blockchain-based approach, as blockchains provide many attractive features, like decentralized architecture and high security assurances. An often neglected aspect of the product life-cycle management is the confidentiality of transactions to hide valuable business information from competitors. To solve the combined problem of trust and confidentiality, we present a confidential licensing and tracking system which works on any publicly verifiable, token-based blockchain that supports tokens of different types representing licenses or attributes of parts. Together with the secure integration of a unique eID in each part, our system provides an efficient, immutable and authenticated transaction log scalable to thousands of transactions per second. With our confidential Token-Based License Management system (cTLM), large industries such as automotive or aviation can license and trace all parts confidentially.

CCS CONCEPTS

• **Security and privacy** → **Digital rights management**; **Privacy-prearving protocols**; *Public key (asymmetric) techniques.*

KEYWORDS

Licensing, Confidential Transactions, Product Life-cycle Management, Track and Trace

Felix Engelmann, Jan Philip Speichert, Ralf God, Frank Kargl, and Christoph Bösch

1 INTRODUCTION

Driven by the current transformation in mechanical part design and manufacturing from product-centric processes to data-centric processes, both opportunities and challenges rise alike. Technologies like additive layer manufacturing (ALM) enable an evolution from legacy part warehouses, logistic centers and just-in-time arrivals to a flexible manufacturing environment in which parts are produced on demand at or near the place of operation. This decoupling of the physical location where a part is either designed or ordered from to the location where it is later manufactured has the potential to lower costs and speed up availability since, e.g., transportation costs and time are reduced to a minimum. However, as it implies the transfer of highly sensitive design, development and production data into an unregulated environment, new challenges in terms of information security, intellectual property (IP) protection, and validation/verification of manufactured parts against the original design arise. In order to establish an end-to-end chain of trust in such an ALM environment, every aspect from the design phase to the materialization of the actual part needs to be secured with an appropriate technology.

The identification of the part itself can be achieved easily by the integration of a tamper resistant electronic identification (eID) feature in the 3D-printed part as shown in [18], for example, by embedding an RFID chip. Such an eID feature is an enabler for identifying original parts, protecting intellectual property rights, granting licenses, and for tracking and tracing of parts in product life-cycle management systems (PLM-systems). A monolithic integration of an eID feature in e.g. a 3D-printed aircraft part provides a high level of physical security, effectively preventing a non-destructive removal of the eID and therefore increasing the protection level against product piracy, fraud, and potential safety risks arising from non-authorized parts.

A real life use-case is an actual Airbus A300/A310 spare part which is only manufactured on demand due to unavailability of the original supplier and lack of legacy manufacturing tools. The so-called belt mold (Figure 1) is part of the flight attendant seat and is currently manufactured with the aviation certified high performance thermoplastic ULTEM™. Previous work [18] already proved the practicability of integrating an eID into the belt mold on an industrial grade 3D-Printer.

However, to realize a fully trusted supply chain it is necessary to establish data continuity from the first design of the 3D model to the actual print job at the print service provider, i.e. the company which operates the 3D printer and materializes the part. Such a trusted supply chain enforces liability on each step and provides auditability. As stated before, the described industrial environment is likely to be inhomogeneous as the print service provider may be able to produce parts for a variety of different industries. While forwarding of actual printing data to printers can be established through traditional IT-systems, like webservers providing the data for authenticated download, the actual management and verification of licenses is more challenging. The use of a blockchain-based approach is an attractive solution, as it omits the necessity of trust-centers and provides a high degree of data consistency and persistence for all participants. The original design of blockchain technology is based on exhaustive transparency of all transactions stored in the
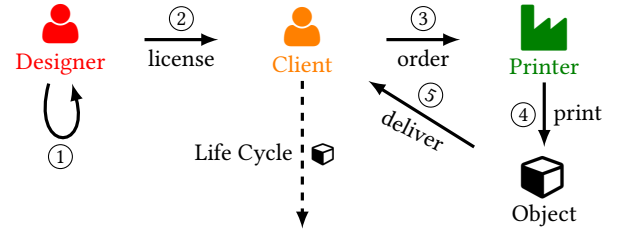

Figure 2: Setting of a license management system with entities and interactions

blockchain in order to ensure their consistency and that data cannot be altered by anyone and unnoticed by the public. In our use-case, this not only allows for a verification whether the printed part matches its design, but also for a licensing system where customer and contractor can handle their order management, adding a further level of security. Together with the unique identification feature of the manufactured parts, our proposed system effectively reduces the risk of product piracy with a strict match between ordered parts and the eIDs registered in the blockchain.

However, this transparency conflicts with the common approach how data is handled and controlled in a company, especially in industries where non-disclosure agreements (NDAs) are widely used in order to protect IP. Storing the licensing data directly on a transparent blockchain violates such agreements and is seen as non-desirable or even prohibitive. To many companies considering adopting blockchain-based systems, this problem is often not even clear during initial planing of projects. To address this, we present our novel confidential Token-Based License Management (cTLM) which allows to satisfy both the requirement for a high level of data security in an open industrial environment as well as the requirement for a high level of privacy. We achieve this licensing and tracking by representing licenses and attributes as tokens of different types. In addition to persons and companies, parts are also identified by accounts in these systems. The possession of a token of some type is then interpreted as having a specific property. All tokens are transferred on a privacy-preserving Blockchain supporting multiple tokens, such as SwapCT [8].

2 USE CASE SETTING

In this section, we describe the general setting in which we apply our license management system shown in Figure 2. First, there is a designer creating a digital representation of a 3D object ①. This 3D file is then registered in our system by the designer, attributing this design to the designer. Once registered, the designer sells licenses either directly to manufacturers ③ or to intermediaries ②. Anyone with a license can manufacture the part and register it by permanently binding the license to a unique identifier assigned to the part ④. Therefore parts are first-class citizens in our transaction system and have their own accounts to which license tokens are transferred. This allows someone handling the part to verify its legitimacy ⑤. Further attributes arising during the part's life cycle, e.g. assembly, disassembly, certification and recycling, are also permanently logged to allow for continuous tracking of the part. This process flow is especially relevant in industries where parts

need to be meticulously documented like aviation. As licensing and production data provides detailed insight and often leads to a competitive advantage, the actions must only be visible to entities involved in handling the parts.

3 CONFIDENTIAL TOKEN TRANSACTION SYSTEM

Our system is built on top of an unspent transaction output (UTXO) protocol for type aware tokens. Generally, a transaction output is represented as a tuple (a, ty, P) of amount a in type ty owned by party P. All three values are hidden inside a one-time account acc. A transaction in such a protocol take the form takes a list of previous transaction outputs which are used as inputs to the transaction. Their tokens are redistributed to a list of outputs. The protocol has to assure that for each type, the sum of input amounts must be equal to the sum of output amounts and the input must belong to the transactor. The protocol additionally has to provide a special transaction to create a new token type ty^* and assign its initial value v^* to an account P^*.

Our System requires two types of long term accounts, one from which tokens are spendable and receive only accounts. The regular long term accounts, e.g. P^*, have a secret key lts with $(\text{lts}, \text{ltv}, \text{ltp}) \xleftarrow{\$} \text{AccGen}()$ and lts is used to authorize spending. Receive only accounts have no secret key and thereby are not be able to sign outgoing transactions $(\text{ltv}, \text{ltp}) \xleftarrow{\$} \text{ItemGen}(\text{eID})$ where eID is any identifier. A recipient retrieves all received tokens, from a read only account or a standard account with $\text{View}(\text{ltv}, \text{acc})$ where ltv is the long term view key available for regular account and receive only account.

3.1 Concrete Scheme

One possible privacy preserving transaction system with confidential types is SwapCT [8]. It achieves sender and receiver anonymity by one-time addresses and stores amounts and types in commitments.

While SwapCT is focused on the conservation of tokens within one transaction, we formalize additional parts of the transaction system. I.e., Verify explicitly verifies a transaction in relation to a state which captures all previous valid transactions. To support receive only accounts, we add an explicit ItemGen algorithm to generate an account without a spend key lts. New token types are added to the system by CoinGen which generates a valid signature σ which is accepted by Verify. The SwapCT Receive algorithm is split into View and Receive to allow viewing of received tokens without requiring the spend key.

Definition 3.1. A privacy-preserving multi-type system based on one-time accounts consists of the following algorithms which are a close adaptation of SwapCT:

$\text{pp} \leftarrow \text{Setup}(1^\lambda)$: takes the security parameter λ and outputs public parameters pp, implicitly given to the subsequent algorithms.

$(\text{lts}, \text{ltv}, \text{ltp}) \leftarrow \text{AccGen}(\text{seed})$: optionally takes a seed, if non provided uses a random seed and outputs a long term spend key lts, view key ltv and the matching public key ltp.

$(\text{ltv}, \text{ltp}) \leftarrow \text{ItemGen}(\text{eID})$: takes a part $\text{eID} \in \{0, 1\}^\lambda$ and outputs a long term view key lts with matching public key ltp.

$(\text{acc}, \text{ck}) \leftarrow \text{OTGen}(\text{ltp}, \text{ty}, a)$: generates a one-time account from a long term public ltp, a type $\text{ty} \in \mathbb{T}$ and an amount a. It returns an account acc and a coin key ck.

$\sigma \leftarrow \text{Spend}(\mathcal{S}, \mathcal{T})$: takes a set of inputs $\mathcal{S} = \{a_i^{\mathcal{S}}, \text{ty}_i^{\mathcal{S}}, \text{ck}_i^{\mathcal{S}}, \text{sk}_i, \text{acc}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}$ with amounts $a_i^{\mathcal{S}}$ of typ $\text{ty}_i^{\mathcal{S}}$ with coin key $\text{ck}_i^{\mathcal{S}}$ and secret key sk_i for the account $\text{acc}_i^{\mathcal{S}}$. The outputs $\mathcal{T} = (\text{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \text{ty}_i^{\mathcal{T}}, \text{acc}_i^{\mathcal{T}})$ are defined by their amount $a_i^{\mathcal{T}}$ of type $\text{ty}_i^{\mathcal{T}}$ hidden by coin key $\text{ck}_i^{\mathcal{T}}$ in the account $\text{acc}_i^{\mathcal{T}}$. The algorithm outputs a signature σ for the transaction tx which is defined as $\text{tx}(\mathcal{S}, \mathcal{T}) := (\{\text{aux}_i^{\mathcal{S}}\}_{i=1}^{\mathcal{S}}, \{\text{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|})$ with some auxiliary information aux_i anonymously referencing the inputs.

$\sigma \leftarrow \text{CoinGen}(\mathcal{T})$: takes a set of outputs defined as $\mathcal{T} = (\text{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \text{ty}_i^{\mathcal{T}}, \text{acc}_i^{\mathcal{T}})$ and outputs a signature σ for the token initiation transaction defined as $\text{tx}(\mathcal{T}) := (\{\}, \{\text{ty}_i^{\mathcal{T}}, \text{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|})$.

$\text{state}'/\perp \leftarrow \text{Verify}(\text{state}, \text{tx}, \sigma)$: takes the state state of the system and a transaction tx with its signature σ. It either outputs a new state state' with the transaction included or $\perp$.

$(a, \text{ty}, \text{ck}) \leftarrow \text{View}(\text{ltv}, \text{acc})$: takes a long term view key ltv for an account acc and outputs the containing amount a of typ ty with coin key ck.

$\text{sk} \leftarrow \text{Receive}(\text{lts}, \text{acc})$: takes a long term secret key lts for an account acc and outputs the corresponding account secret key sk.

$0/1 \leftarrow \text{ChkKey}(\text{sk}, \text{acc})$: takes an account acc and a secret key sk and outputs a bit depending on the validity.

$0/1 \leftarrow \text{ChkVal}(\text{ck}, \text{ty}, a, \text{acc})$: takes an account acc a coin key ck and checks if the ty and amount a match the account.

The system has to fulfill the following correctness criteria:

Definition 3.2 (Correctness). A multi-type transaction system is correct if

correctly generated one-time accounts are viewable: For all $\text{ty} \in \mathbb{T}$ and all $a \in \{0, \dots, 2^{64}-1\}$ and $(\text{lts}, \text{ltv}, \text{ltp}) \leftarrow \text{AccGen}()$ or $(\text{ltv}, \text{ltp}) \leftarrow \text{ItemGen}(\text{eID})$ the account $(\text{acc}, \text{ck}) \leftarrow \text{OTGen}(\text{ltp}, \text{ty}, a)$ is viewable with $(a', \text{ty}', \text{ck}') \leftarrow \text{View}(\text{ltv}, \text{acc})$ such that $(a, \text{ty}, \text{ck}) = (a', \text{ty}', \text{ck}')$.

correctly generated one-time accounts from AccGen is receivable: For all $\text{ty} \in \mathbb{T}$ and all $a \in \{0, \dots, 2^{64} - 1\}$ and $(\text{lts}, \text{ltv}, \text{ltp}) \leftarrow \text{AccGen}()$ the account $(\text{acc}, \text{ck}) \leftarrow \text{OTGen}(\text{ltp}, \text{ty}, a)$ is receivable with $\text{sk} \leftarrow \text{Receive}(\text{lts}, \text{acc})$ such that $\text{ChkKey}(\text{sk}', \text{acc}) = 1$.

honestly generated transactions should verify: For any tuple $\mathcal{S}, \mathcal{T}$ with the structure from Spend, where

- $\forall i \in [|\mathcal{S}|] : \text{ChkKey}(\text{sk}_i, \text{acc}_i^{\mathcal{S}}) = 1$
- $\forall i \in [|\mathcal{S}|] : \text{ChkVal}(\text{ck}_i^{\mathcal{S}}, \text{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \text{acc}_i^{\mathcal{S}}) = 1$
- $\forall i \in [|\mathcal{T}|] : \text{ChkVal}(\text{ck}_i^{\mathcal{T}}, \text{ty}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \text{acc}_i^{\mathcal{T}}) = 1$
- $\forall \text{ty} \in \{\text{ty}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|} : \sum_{\{i:\text{ty}_i^{\mathcal{S}}=\text{ty}\}} a_i^{\mathcal{S}} = \sum_{\{i:\text{ty}_i^{\mathcal{T}}=\text{ty}\}} a_i^{\mathcal{T}}$

holds and all input accounts $\text{acc}_i^{\mathcal{S}}$ are spendable in state, it holds for any proof $\sigma \leftarrow \text{Spend}(\mathcal{S}, \mathcal{T})$ that $\text{Verify}(\text{state}, \text{tx}, \sigma) \neq \perp$ with $\text{tx} = \text{tx}(\mathcal{S}, \mathcal{T})$.

new generated types are valid: For all tuples $\mathcal{T}$ where all $\text{ty}_i^{\mathcal{T}}$ are not previously registered in state and $\forall i \in [|\mathcal{T}|]$: $\text{ChkVal}(\text{ck}_i^{\mathcal{T}}, \text{ty}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \text{acc}_i^{\mathcal{T}}) = 1$, it holds for any $\sigma \leftarrow \text{CoinGen}(\mathcal{T})$ that $\text{Verify}(\text{state}, \text{tx}, \sigma) \neq \perp$ with $\text{tx} = \text{tx}(\mathcal{T})$.

3.2 Construction for missing algorithms

For the algorithms which have no equivalent in SwapCT, we present the construction in this section. Spend is the immediate combination of $\text{Seal}(\text{Offer}(\mathcal{S}, \mathcal{T}))$. To generate a receive only account, we us a key derivation function $\text{KDF} : 0, 1^* \times D \rightarrow \mathbb{K}$ where D is the domain of labels for what the key is used and $\mathbb{K}$ is the appropriate key space. We present ItemGen in Algorithm 1. The tracking and view long term secret keys tsk, vsk are derived from the eID and their corresponding public keys are generated with the public key function PK. Importantly, the spending long term public key pk is directly generated by the KDF such that the private key is infeasible to calculate.

Algorithm 1 ItemGen

Require: eID
  $\text{tsk} \leftarrow \text{KDF}(\text{eID}, \texttt{track})$
  $\text{vsk} \leftarrow \text{KDF}(\text{eID}, \texttt{view})$
  $\text{vpk} \leftarrow \text{PK}(\text{vsk})$
  $\text{tpk} \leftarrow \text{PK}(\text{tsk})$
  $\text{pk} \leftarrow \text{KDF}(\text{eID}, \texttt{own})$
  $\text{ltp} = (\text{vpk}, \text{tpk}, \text{pk})$
  $\text{ltv} = (\text{vsk}, \text{tsk})$ **return** ltv, ltp

---

3.3 Our requirements on a private transaction System

If the above mentioned operations are supported in a privacy-preserving manner, the protocol is suitable for our application. We require the following privacy guarantees from the token protocol, for our system to be secure:

Sender-Anonymity: The owners of transaction inputs must only be known to the transaction creator.

Recipient-Anonymity: The owners of outputs of a transaction must only be known to the sender, i.e. the creator of the transaction.

Confidentiality: The amounts and types of tokens transferred in a transaction must only be known to the sender. Each recipient should only have access to the amount and type of their respective output.

3.4 Generic Operations

As most operations of our proposed system require a transfer of tokens or issuing a new token type, we present high level operations.

Given a state, an amount a for a new type with domain $d \in 0, 1^*$ and pre-image $p \in 0, 1^*$ to the owner of the long term key ltp we issue a token type with Algorithm 2. We use a random oracle $\mathcal{H} : D \times 0, 1^* \rightarrow \mathbb{T}$ from the set of domains D and arbitrary input to a type.

Algorithm 2 Issue Token

Require: $\text{state}, a, d, p, \text{ltp}$
  $\text{ty} \leftarrow \mathcal{H}(d \| p)$
  $(\text{acc}, \text{ck}) \leftarrow \text{OTGen}(\text{ty}, a, \text{ltp})$
  $\sigma \leftarrow \text{CoinGen}(\mathcal{T} = \{(\text{ck}, a, \text{ty}, \text{acc})\})$
      return $\text{Verify}(\text{state}, \text{tx}(\mathcal{T}), \sigma)$

Given a state, an amount a of tokens of typ ty from a one-time account acc with keys $(\text{lts}, \text{ltv}, \text{ltp})$, Algorithm 3 transfers the tokens to the owner of the long term key $\text{ltp}^{\mathcal{T}}$.

Algorithm 3 Transfer Token

Require: $\text{state}, a, \text{ty}, \text{acc}^{\mathcal{S}}, (\text{lts}^{\mathcal{S}}, \text{ltv}^{\mathcal{S}}, \text{ltp}^{\mathcal{S}}), \text{ltp}^{\mathcal{T}}$
  $\text{sk} \leftarrow \text{Receive}(\text{lts}^{\mathcal{S}}, \text{acc}^{\mathcal{S}})$
  $(\text{ty}', s, \text{ck}) \leftarrow \text{View}(\text{ltv}^{\mathcal{S}}, \text{acc}^{\mathcal{S}})$
  if $\text{ty}' \neq \text{ty} \vee s < a$ **then return** $\perp$
  $(\text{acc}^{\mathcal{T}}, \text{ck}^{\mathcal{T}}) \leftarrow \text{OTGen}(\text{ty}, a, \text{ltp}^{\mathcal{T}})$
  $\mathcal{S} = \{(s, \text{ty}, \text{ck}, \text{sk}, \text{acc}^{\mathcal{S}})\}$
  if $s - a = 0$ **then**
      $\sigma \leftarrow \text{Spend}(\mathcal{S}, \mathcal{T} = \{(\text{ck}^{\mathcal{T}}, a, \text{ty}, \text{acc}^{\mathcal{T}})\})$
  else
      $(\text{acc}_R, \text{ck}_R) \leftarrow \text{OTGen}(\text{ty}, s - a, \text{ltp}^{\mathcal{S}})$
      $\mathcal{T} = \{(\text{ck}^{\mathcal{T}}, a, \text{ty}, \text{acc}^{\mathcal{T}}), (\text{ck}_R, s - a, \text{ty}, \text{acc}_R)\}$
      $\sigma \leftarrow \text{Spend}(\mathcal{S}, \mathcal{T})$
  return $\text{Verify}(\text{state}, \text{tx}(\mathcal{S}, \mathcal{T}), \sigma)$

---

4 OUR TOKEN-BASED LICENSING SYSTEM

Given the scenario above and a multi-type confidential transaction protocol as explained before, our system requires to track licenses and the state of objects in a distributed system. The persisted information must be immutable and should be visible only to those parties concerned. To achieve the persistence and immutability without a trusted third party, we use a distributed ledger.

An easy approach for a blockchain based license transaction system is to store the state of the licenses and tracked objects in a smart contract and programmatically update their state according to the smart contract rules. We show that a license management system does not require arbitrary state changes, as is possible with e.g. Ethereum, but can work on a much simpler transaction model. Our approach achieves this using a transaction logic for multiple token types. The benefit of this reduced requirement is an enhanced level of privacy. We propose an application on a blockchain based system to confidentially store the license and life-cycle information of parts which relies on distributed trust and zero knowledge proofs achieving a publicly verifiable history of operations. Therefore, we represent licenses and attributes of objects as tokens. Each 3D model is linked to its own type. Every token of this type is a license. Abstract attributes are types too and the tokens of this type certify that their owner holds the attribute. All tokens are then transferred confidentially. To assign a license or attribute to an object, the according tokens are transferred to the account of the object. Access to the eID of a part allows reconstruction of the part's account for the verification of its attributes. In the remainder of the

work, we use a multi-type system conforming to the requirements of Section 3.

The following sections describe the actions of the participants in more detail. First a new design is created and issued. Then license tokens of this design are traded. Once the part is manufactured, it gets a unique identifier and an account in the system. By transferring a license to the part it is permanently registered to the system. Any further actions in the life-cycle of the part, such as post processing or quality assurance, are then logged by attributes bound to the part's account. The actions are verifiable by anyone with access to the unique identifier.

4.1 Design Issuance

The process starts with a designer creating a 3D object as a CAD file. The designers and engineers or their companies, respectively, own the intellectual property of the object's design. As our goal is to manage licenses of such designs, we have to identify the first and original owner and prevent others from claiming the property rights after the first registration. We identify a design by a cryptographic hash value of its CAD file.

Small adaptations of the CAD file lead to different object identities, but the exact same file always results in the same identity. Automatically assessing the similarity of two designs is difficult and out of scope of our work. Various authors [5, 11, 13, 15] proposed different approaches of similarity measures, all ineffective in a malicious setting due to heuristic approaches. In our setting the designers are malicious entities trying to claim ownership of an existing design of another designer.

The designer derives a new token type

$$F_\square = \mathcal{H}(\text{'design'} \| \texttt{file.cad})$$

and registers it in our system with an initial amount as a new token on the ledger. The designer now owns the initial supply of tokens of type $F_\square$. This initial supply should be chosen large enough because no new tokens of the same type can be generated again. For general parts, we suggest to have an initial supply of e.g. $2^{64} - 1$ which is plentiful. On the other hand, artificial scarcity is achieved by, e.g., issuing only 100 tokens. The designer is now in possession of the newly issued tokens. A transfer requires the matching private key of the designer. For all other participants, the registration process reveals only that someone registered a token type. Everyone can verify that this type was not registered before but gains no additional information.

Algorithm 4 shows the steps to register a design and requires an amount a, by default $a = 2^{64} - 1$, the file and the long term public key of the designer ltp_D generated by AccGen. The domain 'design' is used to separate tokens representing a CAD design from other token types in the system.

Algorithm 4 Issue Design

Require: state, a, `file.cad`, ltp_D
      return $\mathsf{IssueToken}(\text{state}, a, \text{'design'}, \texttt{file.cad}, \mathsf{ltp}_D)$

4.2 Trading

For any registered design, the designer initially owns all license tokens. Each token represents a license that allows the owner to manufacture one physical object. Most of the designers are not manufacturing the objects themselves but rather sell licenses to manufacturing companies, e.g. 3D print services. License tokens are therefore traded for other tokens in the system, most likely (but not limited to) representing fiat currencies.

To exchange tokens of different types when using SwapCT, it provides a swap mechanism which ensures atomic execution. Either both parties receive each other's tokens, or the swap is not performed at all. The exchanged tokens can either be used to manufacture the object or be traded again by performing another swap transaction. Each owner of a license token can use it to manufacture and register a legitimate object of this type.

The representation of a license as a token deviates slightly from a license in most legal systems. Tokens of a specific type are indifferentiable from each other and not batched together in purchases. Another difference is that there might be many intermediary owners of the license token, before it reaches the manufacturer which is uncommon in legal licensing contracts. Given such intermediary owners and due to the sender anonymity, the reconstruction of a transaction path back to the designer requires all previous owners to cooperate. Nevertheless, every participant of the system can verify that the total amount of tokens per type stays constant.

In a multi-type system which supports additional attributes, it is possible to implement an additional attribute which specifies the age of a token. The conservation non-interactive zero-knwoledge proof of each transaction then assures that all output tokens are one epoch older than the inputs. Thereby the token owner can verify the number of intermediaries.

4.3 Manufacturing

A manufacturer owning a license for an object can download the corresponding file from the designer or any shared storage. Our system implicitly protects the integrity of the file by linking it to the token type of the license. To verify the integrity, the file is hashed to a type $F' = \mathcal{H}(\text{'design'} \| \texttt{file.cad})$ and then compared to the token type $F' \overset{?}{=} F_\square$. If the two hash values match, the fetched file is correct and can be printed.

In most cases, the CAD file is fed into a CNC mill or a 3D printer to create the object. After a successful object of the design exists, the manufacturer is trusted to, e.g., either measure (microscopic surface structure patterns, physically unclonable functions) or assign a unique identifier eID to this part. Ideally, this eID is integrated monolithically in the part in order to prevent an undesirable non-destructive extraction or alternation of the eID. A typical representation of an electronic identification feature is a radio-frequency identification (RFID) tag. RFID tags are widely used in a variety of branches like logistics, commerce, and identification documents.

A minimal RFID system consists of a tag and a reader, where both tag and reader can either be a passive or an active component. The most typical setup for our setting is the Active Reader Passive Tag (ARPT) system [14]. The reader transmits an electromagnetic (EM) field which is used to send both signals and energy. The energy from the EM field induces a current in the antenna of the tag. This current

is sufficient enough to power the tag's small integrated circuit (IC). Depending of the programming of the IC, the tag usually sends back a series of numbers, e.g. a simple eID or a cryptographic signature. This allows for long-term, low-cost, and maintenance free identification system as the tag itself does not need any battery supply. From a security perspective, the eID must not be guessable for an object. According to the NIST SP800 [6] publication, this is achieved by a random bit-string of at least 96 bit. More importantly the eID needs to be a unique feature of the part and difficult to copy to another part. For high stake parts, where identification is important, an integrated smart card might be used [19]. The smart card has the benefit of generating a unique random private key and supports authentication challenges to prove the authenticity. This is much harder to copy onto another part as an RFID identifier, but is orders of magnitude more expensive.

From the eID, the manufacturer generates a new read only account $(\mathsf{ltv}, \mathsf{ltp}) = \mathsf{ItemGen}(\mathsf{eID})$. This enables anyone knowing the eID to verify which tokens were sent to the part by $\mathsf{View}(\mathsf{ltv})$. To identify this part as legitimately manufactured, the license token has to be transferred to the part's account. The special feature of such an account, generating an immutable log of properties (received tokens), is the absence of a spend key. Algorithm 5 denotes the steps to register a valid part with an eID, the CAD file and the manufacturer's funding account acc_M along with the long term secret keys $(\mathsf{lts}_M, \mathsf{ltv}_M, \mathsf{ltp}_M)$.

Algorithm 5 Register Item

Require: $\mathsf{state}, \mathsf{eID}, \mathtt{file.cad}, \mathsf{acc}_M, (\mathsf{lts}_M, \mathsf{ltv}_M, \mathsf{ltp}_M)$
  $\mathsf{ty} \leftarrow \mathcal{H}(\mathtt{'design'} \| \mathtt{file.cad})$
  $(\mathsf{ltv}_I, \mathsf{ltp}_I) = \mathsf{ItemGen}(\mathsf{eID})$
  return $\mathsf{TransferToken}(\mathsf{state}, 1, \mathsf{ty}, \mathsf{acc}_M, (\mathsf{lts}_M, \mathsf{ltv}_M, \mathsf{ltp}_M), \mathsf{ltp}_I)$

4.4 Post Processing & Quality Assurance

After manufacturing, parts regularly undergo post-processing and inspections. To attest such processes in the history of the part, the testing entity transfers a token to the part.

These tokens are not license tokens, but behave the same and can therefore use the same underlying confidential transactions. A property of parts is also represented in a new token type. Instead of the hash value of a CAD file, the new type is derived from hashing a unique string, e.g. `"Quality Assured by A. Inc."` in the domain `'property'`. The creator should publish this unique string to allow others to verify if a token really originated from them. The new type F_Q is then registered on the ledger by Algorithm 6 taking the unique string Q, the authorities ltp_Q and a large initial amount a. Every time the company decides to tag a part with this property, they transfer one token of the type to the part's account.

Algorithm 6 Issue Certificate Tokens

Require: $\mathsf{state}, a, Q, \mathsf{ltp}_Q$
      return $\mathsf{IssueToken}(\mathsf{state}, a, \mathtt{'property'}, Q, \mathsf{ltp}_Q)$

The part then has the license token and the F_Q token of the manufacturer. The absence of a private key for the part locks these

tokens forever to the part. Thus, the tokens cannot be secretly transferred to a different part, given the unique identifier cannot be copied or cloned.

This process is repeated, if required, to log multiple properties to the part. On publication of a transfer, the transaction is timestamped persisting the date and time when the part received the property. The transaction to a part's account is not limited to simple binary properties. Small to medium scalar values can be stored by transferring multiple tokens of the same type to the object. Algorithm 7 requires the part eID, the type of certificate Q and the spending key lts_Q of the owner of acc_Q. The amount of tokens a transferred represents the value. An example could be the quality of the part on a scale from one to five, depending on the number of Q tokens received. Instead of scalar values, the number of tokens can be interpreted as a map to strings.

Algorithm 7 Attest Post-Processing

Require: $\mathsf{state}, \mathsf{eID}, Q, a = 1, \mathsf{acc}_Q, (\mathsf{lts}_Q, \mathsf{ltv}_Q, \mathsf{ltp}_Q)$
  $\mathsf{ty} \leftarrow \mathcal{H}(\mathtt{'property'} \| Q)$
  $(\mathsf{ltv}_I, \mathsf{ltp}_I) = \mathsf{ItemGen}(\mathsf{eID})$
  return $\mathsf{TransferToken}(\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}_Q, (\mathsf{lts}_Q, \mathsf{ltv}_Q, \mathsf{ltp}_Q), \mathsf{ltp}_I)$

4.5 Verification

Anyone with physical access to the part can retrieve the eID from the object and derive the account key with the same steps as the manufacturer previously did. With this key, all tokens transferred to the part can be recovered from the public ledger. The license token is checked to verify the original designer, Q tokens attest a correct print and quality check by a known manufacturer and any further properties. This is shown in Algorithm 8 which requires the parts eID and recovers all received tokens. The transactions of the tokens are all timestamped, which allows the reconstruction of the part's history.

Algorithm 8 Item Verification

Require: $\mathsf{state}, \mathsf{eID}$
  $t = \emptyset$
  $(\mathsf{ltv}_I, \mathsf{ltp}_I) = \mathsf{ItemGen}(\mathsf{eID})$
  for all $\mathsf{acc} \in \mathsf{state}$ **do**
      $d \leftarrow \mathsf{Receive}(\mathsf{ltv}_I, \mathsf{acc})$
      if $d \neq \bot$ **then**
          parse d as $(\mathsf{ty}, a, \mathsf{ck})$
          $t = t \cup (\mathsf{ty}, a, \mathsf{ck})$
  return t

All digital systems are only as reliable as the information inserted into it. The interface between the physical world and the immutable ledger has to be trusted. At least our system incentivizes actors to act correctly, as an immutable documentation proves their operations in case of dispute.

For all other participants of the system without the part or the part number (eID), it is impossible to identify which tokens the part possesses. Thereby, the information is available to parties with legitimate interests only.

Our system can be a basis for a complete tracking and tracing system of parts which is often desired by industries, especially in the aviation industry, where a life-long tracking of parts is not only done in production but also during the full life-cycle of a part. This can easily be used to establish a real digital twin not only on system level, but on part level. While there are already several tracking and tracing systems on part level in aviation industry, none of them presents a complete end-to-end solution as our proposed system would.

4.6 Proxy Accounts for Sensitive Parts

For sensitive parts, it might be interesting to limit the scope of which parties have insight into the tokens owned by the part. While the ownership of tokens is directly derived from the eID and every token received by the part is not spendable, we provide an extension where tracking and viewing of tokens is limited to a part owner. In this case, the view secret key ltv is not derived from the eID directly but randomized by the owner by concatenating randomness s. For verification, the owner creates a designated verifier proof that the item received a specific amount of a given type without disclosing possible other tokens of the part. For a designated verifier party with key ltp_V, the part owner creates a zero knowledge proof of knowledge (PoK):

$$\mathsf{PoK}\big(\mathsf{stmt} = (\mathsf{acc}, \mathsf{ltp}_V, \mathsf{ty}, a, \mathsf{eID})\exists\mathsf{wit} = (\mathsf{lts}_V, \mathsf{ck}, s)$$
$$\text{s.t. } \mathsf{ItemGen}(\mathsf{eID}\|s) = (\mathsf{ltv}_I, \mathsf{ltp}_I) \wedge \mathsf{View}(\mathsf{ltv}_I, \mathsf{acc}) = (\mathsf{ck}, \mathsf{ty}, a)$$
$$\vee\, \mathsf{ltp}_V = \mathsf{PK}(\mathsf{lts}_V)\big)$$

Such a proof transcript still reveals the one-time account acc belonging to the part. To improve privacy at the cost of performance, the proxy party creates an anonymous reference with an anonymity set of n accounts $\{\mathsf{acc}_i\}_{i=1}^n$:

$$\mathsf{PoK}\big(\mathsf{stmt} = (\{\mathsf{acc}_i\}_{i=1}^n, \mathsf{ltp}_V, \mathsf{ty}, a)\exists\mathsf{wit} = (\mathsf{lts}_V, j \in \{1, \dots, n\}, \mathsf{ck})$$
$$\text{s.t. } \mathsf{ItemGen}(\mathsf{eID}\|s) = (\mathsf{ltv}_I, \mathsf{ltp}_I) \wedge \mathsf{View}(\mathsf{ltv}_I, \mathsf{acc}_j) = (\mathsf{ck}, \mathsf{ty}, a)$$
$$\vee\, \mathsf{ltp}_V = \mathsf{PK}(\mathsf{lts}_V)\big)$$

This does not reveal to the verifier which transaction output contains the license or property.

4.7 Life cycle

The digital representation of properties in an immutable ledger is especially relevant during the manufacturing process, but also provides multiple benefits during the life-cycle of the part. Parts being resold or refurbished are tracked by a token of a new type. As an example, B. Inc.'s business model is to buy old parts, refurbish them and then sell them again. This is realised with a token of type "refurbished by B. Inc.", which is transferred to the refurbished part. Everyone trusting that this token is really from B. Inc. can then check if a part is genuine and can view the history of production. This is again performed by Algorithm 6 and 7.

The identifier of the type belonging to known companies might be signed in a traditional PKI based system of which the root is controlled by a regulatory authority of the industry sector. E.g., aviation authorities such as the European Union Aviation Safety Agency (EASA) can sign token types from Original Equipment Manufacturer (OEM) or Tier-1 Suppliers in order to verify the originality of parts or services such as a refurbishment.

Once a property token is transferred to a part, it is impossible to take it away due to the lack or a private key. If removing tokens is desired, a simple convention can resolve this, by issuing a secondary token type, which negates the original one. Any part with only the positive token is valid, but if the part also owns the negated one, it is invalid. Once negated, no one can remove the negated token from the part. One example for such a negated token is the revocation of a part. Assume a company's inventory of parts get stolen. In this scenario, the company can issue a revocation token type "revoked by A. Inc." and transfer tokens to the stolen parts. Everyone trusting A. Inc. can check whether the part was revoked. Such revocation tokens have to be carefully verified if they are genuine or not, as they enable a denial of service attack, if such a token is transferred to a still working part.

4.8 Transient Properties

For many tokens owned by parts, it is crucial that they cannot be transferred away from the part to a different account belonging to a different part. This would facilitate the theft of license tokens and certification of subpar parts. However, some life-cycle properties are transient and being passed around between the part and people or machines interacting with it. This can e.g. be used for a per item payment system. Each part processed earns one token which is then used to claim a reward.

For this requirement, participants must be able to transfer tokens away from the object. This is achieved by a secondary account of each item with a spend key $P = \mathsf{AccGen}(\mathsf{eID})$. A set of a tokens of type T is applied to a part eID from some operator with private key lts_O from account acc_O with Algorithm 9.

Algorithm 9 Apply Transient

Require: $\mathsf{state}, \mathsf{eID}, T, a, \mathsf{acc}_O, (\mathsf{lts}_O, \mathsf{ltv}_O, \mathsf{ltp}_O)$
$\quad \mathsf{ty} \leftarrow \mathcal{H}(\text{'attribute'}\|T)$
$\quad (\mathsf{ltv}_I, \mathsf{lts}_I, \mathsf{ltp}_I) = \mathsf{AccGen}(\mathsf{eID})$
$\quad \textbf{return } \mathsf{TransferToken}(\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}_O, (\mathsf{lts}_O, \mathsf{ltv}_O, \mathsf{ltp}_O), \mathsf{ltp}_I)$

All tokens held in this account can be claimed from the object by anyone having access to the eID, the type T, an amount a (mostly $a = 1$) and a recipient operator ltp_O with Algorithm 10. Even after a token was transferred away from the part, it is still possible to verify that the object once owned the property token. The receiving and removal transactions even leave timestamps for both operations.

Algorithm 10 Recover Transient

Require: $\mathsf{state}, \mathsf{eID}, a, \mathsf{acc}_I, \mathsf{ltp}_O$
$\quad (\mathsf{ltv}_I, \mathsf{lts}_I, \mathsf{ltp}_I) = \mathsf{AccGen}(\mathsf{eID})$
$\quad \mathsf{sk} \leftarrow \mathsf{Receive}(\mathsf{lts}_I, \mathsf{acc}_I)$
$\quad (\mathsf{ty}, s, \mathsf{ck}) \leftarrow \mathsf{View}(\mathsf{ltv}_I, \mathsf{acc}_I)$
$\quad \textbf{return } \mathsf{TransferToken}(\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}_I, (\mathsf{lts}_I, \mathsf{ltv}_I, \mathsf{ltp}_I), \mathsf{ltp}_O)$

---

5 SECURITY ANALYSIS

The confidentiality of the transaction data is important for the participants. Transfers of licenses quite always reflect some business interactions, which gives competitors valuable insight into metrics private to a company. The public nature of a blockchain requires all transactions to be public, so that a consensus can be reached. Our approach uses a blockchain which reaches a consensus and allows to verify the conservation rules publicly without revealing the data. This is achieved by using privacy-preserving transactions [8]. Each transaction includes non-interactive zero knowledge (NIZK) proofs to convince other participants of the system that the transactor created the transaction according to the conservation rules. These NIZK proofs are publicly verifiable and are used to reach consensus about the validity of a transaction.

5.1 Attacker Model

We assume all actors of the system are malicious with the exception that they honestly provide inputs from the physical world. Meaning they try to attack the transaction system but do not insert bogus measurements. The requirement for honest interfaces cannot be solved by any purely digital system. However the persistent logging incentivizes all participants to report correct data as they can be held accountable retrospectively.

The verification and persistence of transactions is performed by the entirety of actors following a consensus protocol. We assume that the majority of actors are honest and we therefore model the verification party as honest-but-curious.

5.2 Security Reduction

All operations of our system either create transactions or receive transaction outputs in a way that the security model of the underlying transaction system is taking care of. So regarding anonymity and confidentiality, our system provides the same guarantees as the used privacy-preserving transaction system. An efficient attacker to the security of our system can be directly used to break the security properties of the underlying UTXO transaction system.

To securely deploy our token licensing system, several components require special attention. In the following sections, we elaborate on specific areas of confidentiality, which are important to license transactions and life-cycle tracking.

5.3 Revealing Transactions

In some circumstances, transactors have to reveal past transactions to third parties or authorities. If the underlying ledger uses anonymity sets, where each transactor is hidden in a set of unrelated transactors and too many transactions in a system are publicly known, it gets more and more difficult to stay anonymous within the network of transactions. So revealing some selected transactions to business partners or regulatory entities does not interfere with the anonymity in the system. It is generally in the participants' interest to keep their transactions secret from the general public and especially from competitors.

However, in some industries access to information needs to be granted to certain institutions or authorities. Example use-cases are regular audits to ensure compliance or an investigation following an accident. To achieve this, an escrow key of the involved companies is stored at a safe place. The escrow key cannot be abused to gain insights into transactions of other participants.

If it is sufficient for the auditing entity, the part owners may present zero knowledge proofs of compliance instead of revealing the secret keys. This is achieved equivalently to the proxy accounts from Section 4.6.

5.4 Item eID Enumeration

Another important issue to care about, are eIDs of parts. As they are the seed of the public account keys, they must not be guessable by someone not owning or having access to the part. Otherwise the part's account can be revealed. Also enumeration attacks where parts have consecutive numbers can lead to attacks on the confidentiality of tokens sent to these parts. A reliably non guessable source as of NIST-SP800 [6] is to use 96 bits of uniformly random data. From this, a part account is derived which is not guessable. This keeps the tokens transferred to the part secret to the owners of the part only. As previously mentioned, another option is to integrate a smart card with a randomly generated private key.

It is still important to know that everyone who had access to the part in its history is able to store the eID and track future transactions of the part.

6 PERFORMANCE

To evaluate the performance of the system we take a closer look at the operations required and estimate a number of transactions per second. Operations are defined as the transfer of tokens from one account to another. Going back to the example of the aircraft industry an airplane is made up of around 350,000 [16] to 6 million [3] parts from hundreds of different suppliers and subsuppliers. All those parts could be integrated into our license management system. Airbus and Boeing manufacture around 800 airplanes per year [2]. This equals to around 8,87 operations per second for a company like Airbus. Taking into account other industries like car manufacturing the numbers of operations are significantly higher. A car is made up of around 7,000 [3] to 15,000 [12] parts and 97 Million cars and commercial vehicles were produced in 2017 [17]. If every part is produced based on a token transfer, this results in around 46,000 operations per second. Extending to the multitude of manufacturing sectors where a confidential license management system could be integrated the potential number of operations could be manifold.

To show the applicability of our scheme, we evaluated the performance relevant for different scenarios. The transaction generation for e.g. registering items is most likely performed by the 3D printer. We use a Raspberry Pi first generation to represent the resource constraint device. Figure 3 shows that usual transactions with few inputs and outputs require less than one minute, negligible in relation to the printing duration.

Transaction verification is most likely performed in an on-premise data center with powerful machines. As test payload we used transactions with 2 inputs and 2 outputs, as they will be most frequent in the system. On an dual socket AMD EPYC 7281 Server (64 threads) with Ubuntu 20.04, we achieved a median of 591 transaction verifications per second with a worst case performance of 582.
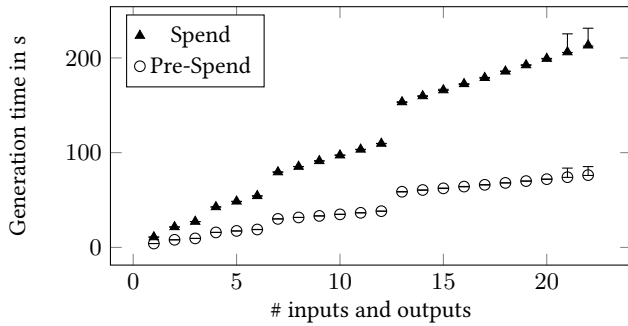
Figure 3: Transaction generation time of our prototype implementation on a Raspberry Pi first generation, representing an embedded system within a printer. As the secret keys are only required in part of the transaction, it is sufficient to execute the Pre-Spend on the embedded device and the remaining part can be offloaded to a powerful computer with minimal privacy loss. We use an anonymity set size of 27. The error bars are the minimum and maximum runtime of 10 executions.

Another relevant time is scanning all transaction outputs for ones belonging to a part or an account. Only with the eID it is possible to detect an output. We measured a median of 260,000 output verifications per second on the above mentioned server.

From these measurements and an estimation of 50,000 operations per second and 5 kB storage per transaction, we derive the computational requirements of our system. Depending on the time by which a transaction output is reused in a subsequent transaction, the verification of transactions can be massively parallelized. For the verification to keep up with the transaction generation, each participant requires around 100 servers to verify all transactions in real time and one server to track new outputs. This is comparable to other systems with similar throughput. To store all transactions, around 20 TB per day are required. An efficient consensus mechanism easily achieves this throughput resulting in a global total order of transactions.

Within a trusted environment, e.g. in a single company, only one node is required to verify and store all transactions. Smaller nodes trusting the central entity can validate part properties based on the central, verified storage. For a system tracking each part of every car manufactured, these requirements are possible with current commodity hardware and allow an immutable, publicly verified log of all operations

6.1 Concurrency

Every state change of a part is reflected with a transaction. However, the order of the persisted transactions cannot be determined in advance. Transacting two tokens to the account of a part at approximately the same instant will result in an arbitrary order. There are no guarantees like first in first out, even for a single part, as each transaction is hiding the recipient and there might be a separate actor transmitting a token to said part. To implement strict conditional dependencies, it is necessary to query the persisted log, if the condition is persisted.

The certification accounts have to split their initial output, after issuing, into multiple smaller outputs. Using the change output from the previous part is only possible after the transaction is persisted, so repetitively using the change to attest many parts in short succession is not possible. This issue is mitigated by using smaller outputs in a round robin fashion.

7 RELATED WORK

As our system combines privacy-preserving blockchain protocols with blockchain based license management, we present recent research results from both areas.

7.1 Privacy Perserving Blockchains

Cloaked Assets (ZkVM). The team of Stellar[1] introduces a zero knowledge virtual machine [4], which executes smart contracts in a privacy-preserving manner. This enables management of typed tokens and provides the necessary functionality to be used as a basis for our application. We chose SwapCT as Cloaked Assets do not provide sender and receiver anonymity.

Conficential Assets on MimbleWimble. Yi Zheng et al. [20] extended the MimbleWimble protocol with confidential assets to support multiple tokens in a confidential way. This chain may be used as a basis for our system with the limitation, that transactions are created interactively with the recipients. The aggregation characteristics of MimbleWimble are highly favourable for high throughput applications at the cost of reduced privacy.

7.2 Blockchain PLM/DRM

Multiple authors have described the use of blockchain technology for product life-cycle management [1, 7, 9]. Unfortunately, all previously presented solutions neglect the crucial confidentiality of operations. With all transactions in public, valuable business metrics are analysed by everyone and may be used for harm. The alternative to deploy a private ledger in a consortium does not solve the confidentiality requirement, as the very competitors, to who insight is most valuable, still have access as they most likely belong to the same consortium. Our solution maintains the confidentiality and is therefore also suitable for private consortia as well as public ledgers, depending on the participants and the public interest.

Herbert and Litchfield [10] propose two blockchain based software license managements. The first uses additional semantics of standard bitcoin transactions, providing no confidentiality. In their second scheme, the vendor is trusted to manage encrypted licenses persisted within transactions. While improving data protection, they require an honest entity to achieve the same confidentiality as our protocol.

8 CONCLUSION

The tracking and tracing of parts in complex industries such as automotive or aviation is a fundamental task, especially having intricate business relations. In addition, reliably differentiating legitimately licensed parts from reputable sources is a difficult feat. On top, the information about parts must remain secret from competitors and alike.

---

[1]https://www.stellar.org/

Our proposed system is able to provide all required functionality while maintaining confidentiality of operations in an untrusted environment. We defined protocols to represent the complex life-cycle actions in the form of basic transfers of tokens. By building the system on top of a privacy-preserving multi token blockchain, our system inherits the confidentiality guarantees. Given an efficient consensus mechanism for transaction ordering, the verification is massively parallelizable and can handle up to 10 transactions per core per second. These features combined with a decent performance makes our system directly applicable to many real world life-cycle management scenarios.

ACKNOWLEDGEMENTS

REFERENCES

[1] Saveen A Abeyratne and Radmehr P Monfared. 2016. Blockchain ready manufacturing supply chain using distributed ledger. (2016).
[2] Airbus. 2019. *Airbus achieves new commercial aircraft delivery record in 2018.* https://www.airbus.com/newsroom/press-releases/en/2019/01/airbus-achieves-new-commercial-aircraft-delivery-record-in-2018.html
[3] Hans-Henrich Altfeld. 2016. *Commercial aircraft projects: Managing the development of highly complex products.* Routledge.
[4] Oleg Andreev, Bob Glickstein, Vicki Niu, Tess Rinearson, Debnil Sur, and Cathie Yun. 2019. *ZkVM: fast, private, flexible blockchain contracts.* Technical Report.
[5] Xin Chen, Jingbin Hao, Hao Liu, Zhengtong Han, and Shengping Ye. 2017. Research on Similarity Measurements of 3D Models Based on Skeleton Trees. *Computers* 6, 2 (2017), 17.
[6] Morris J Dworkin. 2007. *Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC.* Technical Report.
[7] Felix Engelmann, Martin Holland, Christopher Nigischer, and Josip Stjepandić. 2018. Intellectual property protection and licensing of 3D print with blockchain technology. In *Transdisciplinary Engineering Methods for Social Innovation of Industry 4.0: Proceedings of the 25th ISPE Inc. International Conference on Transdisciplinary Engineering, July 3–6, 2018*, Vol. 7. IOS Press, 103.
[8] Felix Engelmann, Lukas Müller, Andreas Peter, Frank Kargl, and Christoph Bösch. 2021. SwapCT: Swap Confidential Transactions for Privacy-Preserving Multi-Token Exchanges. Cryptology ePrint Archive, Report 2021/631. (2021). https://eprint.iacr.org/2021/631.
[9] Dominik Heber, Marco Groll, et al. 2017. Towards a digital twin: How the blockchain can foster E/E-traceability in consideration of model-based systems engineering. In *DS 87-3 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 3: Product, Services and Systems Design, Vancouver, Canada, 21-25.08. 2017.* 321–330.
[10] Jeff Herbert and Alan Litchfield. 2015. A novel method for decentralised peer-to-peer software license validation using cryptocurrency blockchain technology. In *Proceedings of the 38th Australasian computer science conference (ACSC 2015)*, Vol. 27. 30.
[11] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L Kunii. 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* ACM, 203–212.
[12] Thomas H Klier and James M Rubenstein. 2008. Who really made your car? *Employment Research Newsletter* 15, 2 (2008), 1.
[13] Wenlong Lu, Yuchu Qin, Qunfen Qi, Wenhan Zeng, Yanru Zhong, Xiaojun Liu, and Xiangqian Jiang. 2016. Selecting a semantic similarity measure for concepts in two different CAD model data ontologies. *Advanced Engineering Informatics* 30, 3 (2016), 449–466.
[14] Saeed Manaffam and Amirhossein Jabalameli. 2016. RF-localize: An RFID-based localization algorithm for Internet-of-Things. In *2016 Annual IEEE Systems Conference (SysCon).* IEEE, 1–5.
[15] Duhwan Mun and Karthik Ramani. 2011. Knowledge-based part similarity measurement utilizing ontology and multi-criteria decision making technique. *Advanced Engineering Informatics* 25, 2 (2011), 119–130.
[16] Allison Linn NBC News. 2010. *Hundreds of suppliers, one Boing 737 airplane.* http://www.nbcnews.com/id/36507420/ns/business-us_business/t/hundreds-suppliers-one-boeing-airplane/#.XTmO4ZMzZaQ
[17] International Organization of Motor Vehicle Manufacturers. 2018. *2017 Production statistics.* http://www.oica.net/category/production-statistics/2017-statistics/
[18] A Pagel, Jan Philip Speichert, and Ralf God. 2018. Wie sicher ist der 3D-Druck? *Konstruktionspraxis* 2018, 10 (2018), 34–46.
[19] Wolfgang Rankl and Wolfgang Effing. 2004. *Smart card handbook.* John Wiley & Sons.
[20] Yi Zheng, Howard Ye, Patrick Dai, Tongcheng Sun, and Vladislav Gelfer. 2019. Confidential Assets on MimbleWimble. Cryptology ePrint Archive, Report 2019/1435. https://eprint.iacr.org/2019/1435.