



Contents lists available at ScienceDirect

Information Processing and Management

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



EtherTwin: Blockchain-based Secure Digital Twin Information Management

Benedikt Putz^{*}, Marietheres Dietz, Philip Empl, Günther Pernul

Universitätsstraße 31, Regensburg 93051, Germany



ARTICLE INFO

Keywords:

Distributed ledgers
Blockchain
Digital twin
Industry 4.0
Decentralized application

ABSTRACT

Digital Twins are complex digital representations of assets that are used by a variety of organizations across the Industry 4.0 value chain. As the digitization of industrial processes advances, Digital Twins will become widespread. As a result, there is a need to develop new secure data sharing models for a complex ecosystem of interacting Digital Twins and lifecycle parties. Decentralized Applications are uniquely suited to address these sharing challenges while ensuring availability, integrity and confidentiality. They rely on distributed ledgers and decentralized databases for data storage and processing, avoiding single points of trust. To tackle the need for decentralized sharing of Digital Twin data, this work proposes an owner-centric decentralized sharing model. A formal access control model addresses integrity and confidentiality aspects based on Digital Twin components and lifecycle requirements. With our prototypical implementation EtherTwin we show how to overcome the numerous implementation challenges associated with fully decentralized data sharing, enabling management of Digital Twin components and their associated information. For validation, the prototype is evaluated based on an industry use case and semi-structured expert interviews.

1. Introduction

Industrial control systems (ICS) such as supervisory control and data acquisition (SCADA) systems, human machine interfaces (HMI), programmable logic controllers (PLCs) and other field devices are able to control physical processes within industrial environments. Traditionally, they form the core of industrial infrastructures. In the course of the Industry 4.0, however, these industrial environments further converge with information technology [Rubio, Roman, and López \(2017\)](#). For instance, sensors measuring the conditions of the respective physical processes to control are increasingly installed. This sensor data as well as the ICS systems are integrated to corporate IT systems in order to centrally analyze and manage information about the industrial environment.

The Digital Twin (DT) presents one of the key concepts reflected in the Industry 4.0 movement. In Industry 4.0, the DT can generally be defined as a digital representation of an industrial asset over its entire lifecycle [Boschert, Heinrich, and Rosen \(2018\)](#). To represent and to further monitor its counterpart, the DT incorporates all kinds of asset-relevant information. This includes a multitude of generated sensor data from Industry 4.0 assets, which are united in DTs. Depending on the underlying asset, different lifecycles are covered by the digital twin. From this follows that different participants involved in the lifecycle might provide information for the DT

* Corresponding author.

E-mail addresses: benedikt.putz@ur.de, benedikt.putz@wiwi.uni-regensburg.de (B. Putz), marietheres.dietz@ur.de (M. Dietz), philip.empl@ur.de (P. Empl), guenther.pernul@ur.de (G. Pernul).

Table 1

Comparison of blockchain-based DT-related approaches by considering organizational as well as implementation characteristics. ○ not considered, ● partially considered, ● fully implemented.

	Huang et al. (2020)	Hasan et al. (2020)	Angrish et al. (2018)	Dietz et al. (2019)
DT definition	product	any asset	machine events	any asset
Components	○	○	○	○
Lifecycle phases	early & medium	early	medium	early & medium
BC suitability	○	○	○	●
Implementation	○	●	●	○
Open Source	○	●	○	○
Blockchain	unknown	Ethereum	Ethereum	unknown
Off-chain storage	○	●	●	●
Encryption	●	○	○	●
Access control	○	○	●	●
User Interface	○	○	○	●

or need to gather data managed by the DT (*DT data sharing*) Dietz and Pernul (2020a).

To achieve information management and sharing in Industry 4.0 with DTs, some obstacles arise. To manage DT data, involved lifecycle parties need access to the DT. Although the different parties participating in these processes work together, they each pursue different goals. Consider the lifecycle parties involved in an industrial plant, where a DT incorporates all relevant data. The manufacturer of the plant's motors should not gain access to the data about the plant's current status, but should get feedback whenever the motor is maintained in order to optimize the motor's construction and enhance its manufacturing process. In contrast, the maintainer of the plant's motors should only get access to the motor's current status and the components the maintainer is not responsible for, but not to any other component's status of the plant. Thus, the trust when sharing data via the DT is not given per default Malakuti and Grüner (2018). As a result, confidentiality and access control issues arise Dietz and Pernul (2020b). These issues cannot be resolved with a centralized authority, especially in multi-tenant and large-scale environments Esposito, Tamburis, Su, and Choi (2020).

This work addresses the lack of trust and security among multiple parties in DT data sharing by focusing on the following research question:

RQ1. How can the data of Digital Twins be shared among multiple untrusted lifecycle parties while ensuring confidentiality, integrity and availability?

Blockchains and their smart contracts possess various characteristics that can support the security of data sharing Berdik, Otoom, Schmidt, Porter, and Jararweh (2021). For instance, single and multi-party authentication can be implemented in a decentralized way Khan and Salah (2018) – without requiring trust in a central party. Moreover, blockchain solutions enable decentralized management of an asset's lifecycle and supply chain Khan and Salah (2018). Blockchain solutions rely on Decentralized Applications (DApps), user-friendly web-based interfaces to interact with blockchains and their smart contracts. These characteristics offer a novel opportunity to solve the aforementioned obstacles in DT information management.

In this work, we show why a blockchain-based solution is suitable for DT data sharing and propose a blockchain-based information management solution for the DT and the involved lifecycle parties. We go beyond the state-of-the-art research by including DT components with fine-grained access control and providing scalability for sensor data sharing. Finally, our approach is evaluated with a DApp prototype implementation (EtherTwin), an industry use case, expert interviews as well as performance and cost measurements.

The remainder of this work is organized as follows. We introduce related work in Chapter 2. The background of our research is laid in Chapter 3. Afterwards, we outline the logical design of our concept in Chapter 4. Chapter 5 describes the implementation of our EtherTwin DApp, which is subsequently evaluated in Chapter 6. Chapter 7 discusses our prototype in respect to the evaluation and future work. Finally, a conclusion is drawn in Chapter 8.

2. Related work

As DT research began to grow only during recent years, current works mainly propose theoretical frameworks. To date, various works mention the issue of the DT requiring strong security Kaur, Mishra, and Maheshwari (2020); Rubio et al. (2017); Uhlemann, Lehmann, and Steinhilper (2017), however applicable solutions are not provided yet. Especially, the secure management of DT data storage and exchange is important for practical use Malakuti and Grüner (2018).

There have been few other works exploring the blockchain-based accompaniment of assets in supply chain processes with DTs Mandolla, Petruzzelli, Percoco, and Urbinati (2019) and smart objects Meroni and Plebani (2018). Still, a comprehensive implementation of decentralized and secure data sharing for DTs is missing. Moreover, past works have shown the feasibility and advantages of blockchain-based access control for decentralized data sharing Di Francesco Maesa, Mori, and Ricci (2019); López-Pintado, Dumas, García-Bañuelos, and Weber (2019). However, there is no blockchain-based access control model tailored to the requirements of the DT lifecycle.

In the following, we compare previous works that focused on blockchain-based data management in connection with the DT. Table 1 summarizes the comparison by considering organizational aspects of data management as well as implementation characteristics: The first few characteristics are of organizational nature, the following are implementation-related.

Table 2

General lifecycle characteristics (involved parties and data) of an industrial asset. Potentially involved lifecycle parties are highlighted in *italic*.

	Early Phases	Medium Phases	Later Phases
Lifecycle phases	Idea, Planning, Manufacturing	Operation, Maintenance	Demolition, End of Existence
Accruing Data	<ul style="list-style-type: none"> • Sketches • Blueprints • Manuals • Design models 	<ul style="list-style-type: none"> • Sensor data • System logs • Maintenance reports • Simulations 	<ul style="list-style-type: none"> • Condition of the components • Component's location
Involved parties	Owner, Manufacturers, Distributors	Owner, <i>Manufacturers, Distributors</i> , Maintainers	Owner, <i>Manufacturers, Distributors</i> , Maintainers

So far, few works have tackled blockchain-based data management in connection with DTs. Angrish et al. develop a prototype for a peer-to-peer network of manufacturing nodes [Angrish, Craver, Hasan, and Starly \(2018\)](#). Hasan et al. propose a blockchain-based data management approach for the DT creation process [Hasan et al. \(2020\)](#). Huang et al. [Huang, Wang, Yan, and Fang \(2020\)](#) propose a management approach to store all relevant DT data on a custom blockchain. Dietz et al. propose a conceptual approach for blockchain-based DT data management [Dietz, Putz, and Pernul \(2019\)](#).

Thereby, the organizational aspects of the related works vary. While Huang et al. consider the DT being a product [Huang et al. \(2020\)](#), Angrish et al. define the DT as a mere collection of machine events [Angrish et al. \(2018\)](#). The majority of the works, as well as our work, see the DT as a representation of any asset [Dietz et al. \(2019\); Hasan et al. \(2020\)](#), be it a system, product or another physical object. Moreover, so far, none of the related works have tackled the DT as being a complex representation of an asset with sub-components. In contrast, we include components of the DT in our data model. In terms of lifecycle phases (cf. [Table 2](#)), we are the first to consider the DT management as beneficial for later lifecycle phases as well. Works to date have tackled either early [Hasan et al. \(2020\)](#), medium [Angrish et al. \(2018\)](#) or both of these phases [Dietz et al. \(2019\); Huang et al. \(2020\)](#). Additionally, we are the first to fully investigate the suitability of blockchains for DT data management by following a research methodology, while other works either neglect this aspect [Angrish et al. \(2018\); Hasan et al. \(2020\); Huang et al. \(2020\)](#) or only mention, but do not describe a method [Dietz et al. \(2019\)](#).

To date, prototypical implementations have been either neglected [Dietz et al. \(2019\); Huang et al. \(2020\)](#) or only partially accomplished [Angrish et al. \(2018\); Hasan et al. \(2020\)](#). Our work is the first to fully implement a proposed DT data sharing approach. Next to our work, only one other work has made the implementation open source [Hasan et al. \(2020\)](#). All works with an implementation part, however, make use of the Ethereum blockchain. In terms of off-chain storage, related work either do not suggest using it [Huang et al. \(2020\)](#), or suggest to use off-chain storage, but do not implement this part [Angrish et al. \(2018\); Dietz et al. \(2019\); Hasan et al. \(2020\)](#). In our EtherTwin prototype, a fully implemented off-chain storage is present. Encryption is proposed in two of the four related works [Dietz et al. \(2019\); Huang et al. \(2020\)](#), but is not described in detail and implemented – in comparison to our work. Likewise, access control mechanisms are mentioned in two works [Angrish et al. \(2018\); Dietz et al. \(2019\)](#) but are also not implemented. A user interface is only suggested by a single work [Dietz et al. \(2019\)](#), but we are the first to design and implement one.

The present work develops a component-based data model and an access control model for common lifecycle participants. To summarize, we contribute to DT and blockchain research by providing:

- **fine-grained access control** for DT data sharing in a decentralized setting without a trusted third party (TTP), ensuring confidentiality through encryption
- full-featured **open source prototype** EtherTwin based on blockchain design patterns and state of the art DApp technologies (Ethereum, Swarm) with performance/cost measurements
- evaluation based on an **industry use case** and expert interviews

3. Background

The background of this work is divided into three sections. [Section 3.1](#) describes the foundations of DT research. Subsequently, the background of DApps is laid in [Section 3.2](#).

3.1. Digital twins

The DT is an emerging paradigm focusing on an enterprise asset – usually, a system, product or process, along its lifecycle [Boschert et al. \(2018\)](#). Its core goal is to virtually represent this asset as close to reality as possible [Boschert et al. \(2018\)](#). The lifecycle phases covered by a DT strongly depend upon its corresponding asset. Nevertheless, common early phases are *Idea, Planning* and *Design*, while an asset's *Operation* can be considered one of the medium phases and the asset's *Demolition* is one of the final phases [Dietz and Pernul \(2020b\)](#). Thereby, each phase can span many years. For instance, planning a complex asset like global satellite networks could take up to 10 years until *Operation*, while some legal regulations may command to safely store the asset after its decommission. Especially, these long and safety-oriented lifecycle phases require a tamper-proof data storage solution. By including various data sources and by

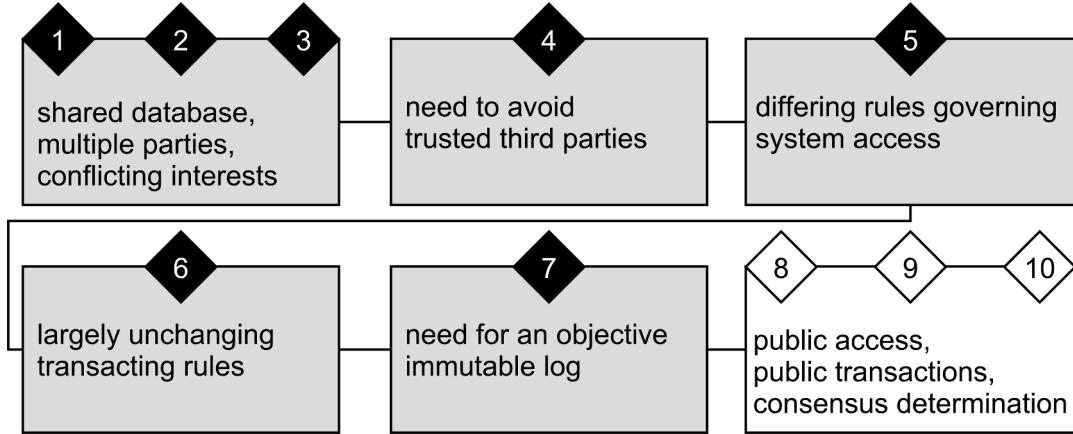


Fig. 1. Blockchain decision path by Pedersen et al. [Pedersen et al. \(2019\)](#).

integrating the multiple parties involved in these lifecycle phases, the DT unifies asset-specific data from previously separated domains [Ríos, Hernández, Oliva, and Mas \(2015\)](#). For instance, the asset's composition, sensor data of the asset's environment as well as simulation models can be included centrally in a DT [Dietz and Pernul \(2020a\)](#). This further promotes the complete traceability of assets and their components, especially if the assets (e.g. industrial plants, cars) comprise components from several manufacturers. Thereby, feedback loops across different lifecycle phases can be realized that support the concerned lifecycle parties when looking for improvement of their components [Boschert et al. \(2018\)](#). For instance, manufacturers can get insights from the operational phase of the asset and draw conclusions about the effectiveness of their components.

For the remainder of this work, we put the person that owns the physical asset in the role of the Owner. Whenever this kind of Owner is meant, it is written in capitals. We further claim that the ownership of the physical asset implies the ownership of the digital twin. Otherwise, two different parties respectively owning either one of them would commonly not trust each other. Thus, the interaction of digital twin and physical twin would not be achieved.

[Table 2](#) summarizes the common lifecycle phases and points out the potentially involved lifecycle parties and the accruing data in the respective phases. Note that the data is continuously transformed along the lifecycle phases. For example, sketches of an industrial asset might exist from the *Idea* phase, transform into a blueprint in the *Design* phase. Also, design models might be created in the *Design* phase and elaborated towards fully-fledged simulations in the *Operation* and *Maintenance* phase. In terms of the involved parties, italicized parties are only potentially involved. For instance, consider the Owner sketching the asset during the *Idea* phase. Afterwards, the manufacturer elaborates this sketch towards a blueprint (*Design*) and manufactures the asset (*Manufacturing*). Later on, the Owner commissions the maintainer to put the asset into *Operation*.

Nevertheless, there are still some obstacles to overcome. Commonly, an industrial asset represents a complex system, product or process. As a consequence, a multitude of parties are involved. Consider an industrial plant consisting of various ICSs. Each of these systems potentially has its own manufacturer and in business life, they might be competitors. This leads to enormous trust issues, and towards current practices of each lifecycle party building their own DT [Malakuti and Grüner \(2018\)](#). Meanwhile, this practice contradicts the very idea of DTs. Furthermore, it results in the disappearance of the DT's core benefits like feedback loops to other lifecycle phases and parties. To overcome current malpractices and to motivate users to share their data among parties with different trust levels, our research aims to provide a strong platform with sufficient security (i.e. access control mechanisms) among untrusted parties.

3.2. Blockchain and decentralized applications

To address the complex issues of the DT sharing ecosystem, we investigate if blockchain technology is suitable. Pedersen et al. propose a ten-step decision path to determine if blockchain is a good fit [Pedersen, Risius, and Beck \(2019\)](#). The ten requirements are outlined in [Fig. 1](#). For the DT lifecycle, there are multiple parties with the need for a shared database, which may have conflicting interests and thus, varying trust levels (*steps 1–3*). While in theory the lifecycle parties could rely on a TTP service, the dynamics and variety of DT data sharing hamper the management through a TTP. As [Table 2](#) highlights, various data and data types are involved with varying velocity and integrity requirements. Integrity of stored data is an especially important security concern in IoT environments [Zhao, Chen, Liu, Baker, and Zhang \(2020\)](#). A TTP represents a single point of failure and an attack could interfere with the integrity of the data, making it preferable to avoid third parties [Li, Wu, Jiang, and Srikanthan \(2020\)](#), which suggests that it could be a good fit for our work. (*step 4*). Moreover, the participants of the lifecycle require different access privileges depending on their role and characteristics, which means there are differing rules governing system access (*step 5*). Although system access rules differ in practice, the rules of transacting with DT data do not change frequently (*step 6*). The blockchain's immutable log is helpful to ensure integrity and traceability of all

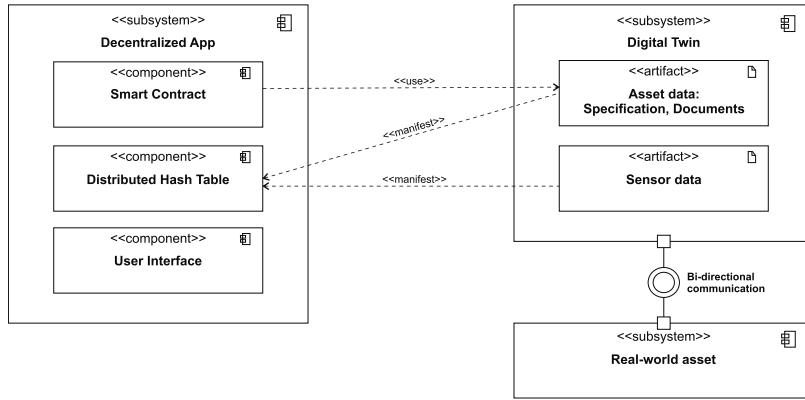


Fig. 2. Component Diagram describing the Digital Twin Sharing Context.

changes to the DT data, for example, in case of security issues or malfunctions. Furthermore, the documentation of changes made to data items is required to meet compliance requirements for some DTs (*step 7*). The need for public access largely depends on the DT's underlying asset and industry (*steps 8 - 10*). We do not make an assumption in this regard and design our application to work with both permissionless and permissioned networks.

After determining that blockchain is suitable, we explain the software components required for building a DT information management application. DApps are a new paradigm for developing distributed applications Xu, Weber, and Staples (2019). Application logic is fully decentralized, since front end code runs in the user's browser and back end code runs in smart contracts on the blockchain nodes. Decentralization comes with the advantage of full transparency of the application code as well as auditability of changes to a smart contract state. Dynamic smart contract access control models can be used to authorize state changes Di Francesco Maesa et al. (2019).

Full replication of blockchain data necessitates storing complex data elsewhere Baig and Wang (2019), leading to the concept of off-chain storage. A common approach is to use Distributed Hash Tables (DHTs), since they fit the decentralized paradigm well. Data items are content-addressed and replicated within the network based on a routing layer. Modern DHTs such as Swarm¹ are based on the established and secure DHT routing technology S/Kademlia Baumgart and Mies (2007) and integrate well with blockchains such as Ethereum².

Blockchain smart contracts also need to ensure sufficient access control to prevent unauthorized modification of smart contract state. Numerous authors have developed access control concepts based on smart contracts. These are based on the existing access control models role-based (RBAC) Cruz, Kaji, and Yanai (2018) or attribute-based access control (ABAC) Rouhani, Belchior, Cruz, and Deters (2020), but there are also proposals for ciphertext-policy attribute-based encryption Badsha, Vakilinia, and Sengupta (2020). Zhang et al. present an access control framework for the Internet of Things (IoT) supporting flexible access control methods Zhang, Kasahara, Shen, Jiang, and Wan (2019). Rouhani et al. also provide a comprehensive overview of smart contract based access control approaches Rouhani et al. (2020).

4. System model

The following sections describe the logical structure of our DApp. Section 4.1 provides an overview of the DApp's entities. Section 4.2 explains the twin and its subparts, while Section 4.3 focuses on the authorization of participating parties.

4.1. Overview

To capture context, the component diagram in Fig. 2 provides an overview of the DT sharing approach. In our system model, a component diagram defines physical as well as logical components and their dependencies. Therefore, it is well suited to put software architectures like our DApp into context.

Fig. 2 illustrates the connection between real-world asset, DT and the developed DApp. These components represent a greater architectural unit (*subsystems*). The first two subsystems present the sole DT paradigm, consisting of the DT and its real-world asset connected by the bi-directional communication *interface*. One of the two *artifacts* within the DT is asset data, e.g. the specification of the asset with its compositional structure and documents about the asset. The other artifact is the sensor data produced in the asset's environment. To enable data sharing, the DApp is added. It includes the *components* Smart Contract, DHT and User Interface. The *dependency* relations show the association of the DT data to the DApp. For instance, the Smart Contract requires the specification data of the asset in order to be built (*usage dependency*). Moreover, the shared DT data is stored in the DHT: The *manifest dependency* shows

¹ swarm.ethereum.org

² ethereum.org

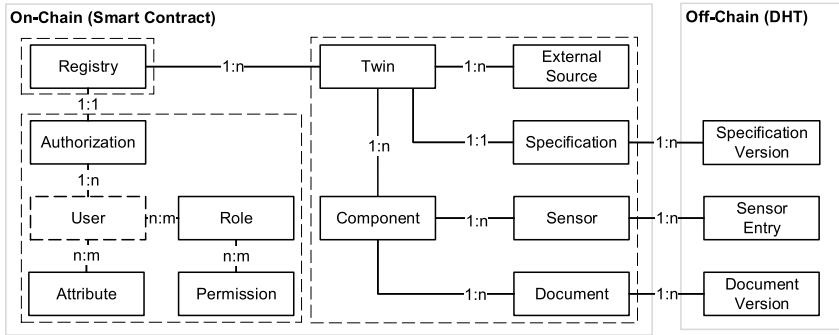


Fig. 3. Entity Relationship Model of the DApp.

that the logical DT artifacts physically manifest in the DHT of our proposed DApp. Finally, the user interface component provides access to the data for all participating lifecycle parties.

4.2. Entity relationship model

Fig. 3 illustrates entities and relationships for DT sharing with a DApp. The dashed borders show a logical grouping into three main components: *Registry*, *Authorization* and *Twin Data*. To keep track of all available twins, a single access point is needed, referred to as the *Registry*. Similarly, the *Authorization* group of entities represents the access control model, which is explained in detail in Section 4.3. *Twin Data* is derived from DT sharing requirements elaborated in our previous work Dietz et al. (2019).

The on-chain entities contain metadata about the DT. The main entity of a DT is the *Specification*, comprising the *Components* of the real-world asset it is representing. *Sensors* and other data (abstracted with the term *Document*) are managed by associating them with the corresponding component. Moreover, for each DT *External Sources* such as legacy systems can be integrated. These can provide additional data to the already incorporated documents and sensor feeds.

Off-chain entities (*Specification Version*, *Sensor Entries*, *Document Version*) contain full data and are linked to on-chain entities, as indicated by 1:n relationships in Fig. 3.

4.3. Access control

In order to share data securely, an authorization and access control policy is required. This way access to data items can be restricted to certain parties. For instance, a maintenance report of an asset's component (e.g. of a PLC) should only be shared with the lifecycle parties of this component (e.g. the PLC's manufacturer). Access control addresses this need by restricting the user operations for data objects. In our approach we follow a hybrid access control model, combining RBAC and ABAC. While a role refers to a certain organizational function, a particular attribute refers to a specific characteristic of a user. During the DT lifecycle, each user interacts with certain twin components, which constitute the user's attributes in our model. While roles are predefined, these attributes allow access control on-the-fly.

Our proposed approach is modeled after the RBAC-A (role-centric) combination strategy, where attributes are applied to constrain RBAC Coyne and Weil (2013); Kuhn, Coyne, and Weil (2010). Thereby, the user's assigned role defines the base permissions, while the user's additional attributes can further limit these permissions. The exclusive use of ABAC would create an unnecessary overhead of rules, which control the access of the user. This would further increase complexity, both in terms of attribute combination for the user and the subsequent access granting decisions. Our hybrid access control model for DT data sharing upholds essential RBAC advantages (e.g. ease of user provisioning) and enhances flexibility by integrating attributes.

To provide a profound basis for later implementation, we elaborate a formalism of the access control used in our DT sharing approach. Italicized terms refer to entities from Fig. 3. Every sharing party is considered a $User U := \{u_1, \dots, u_n\}$. In our hybrid access control approach, each user can have one $Role R := \{r_1, \dots, r_n\}$ as well as several $Attributes A := \{a_1, \dots, a_n\}$ per DT. $Components C := \{c_1, \dots, c_n\}$ serve a special purpose in this access control model, as they are used for modeling $Attributes: A := a_1, \dots, a_n | a_i = c_1 \vee \dots \vee c_n$.

To continue, $Permissions P := \{p_1, \dots, p_n\}$ are mainly derived from the user's role but also from its attribute(s). This underlines the hybrid RBAC-A mode in a role-centric realization Kuhn et al. (2010): Roles determine the basic permissions, while some of these permissions are limited by the users' attribute(s). The permissions usually specify the access to an object $O := \{o_1, \dots, o_n\}$ and the allowed operation $Op := \{op_1, \dots, op_n\}$. Objects are always associated to a component to link the asset-relevant data to the component they belong: $o \rightarrow c | o \in O \wedge c \in C$. This results in the following definition of Permissions: $P = Op \times O$, whereby it can be concluded that $p \rightarrow c$.

The n-m relation of users to roles is expressed by $UR = U \times R$. Likewise, the user to attributes relation can be described as $UA = U \times$

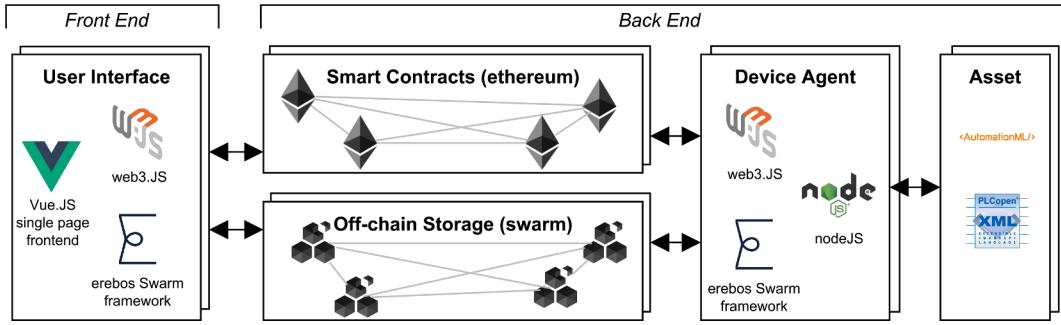


Fig. 4. Technologies used in our prototypical implementation of DT data sharing.

A. Mapping the role-attribute combination to a user results in:

$$\text{assign_users}(r, a) = u \in U \mid (u, r) \in UR \wedge (u, a) \in UA$$

Thereby, the mapping M^{UR} describes the set of actual assignments of users to roles, while M^{UA} determines the set of assigned attributes to users. Similar to the mapping above, the m-n relation between permissions and roles are specified by $PR = P \times R$. Finally, permissions are mapped to role-attribute combinations while the attribute restricts the role permission and M^{PR} describes the set of actual assignments of users to permissions:

$$\text{assign_permissions}(r, a) = p \in P \mid (p, r) \in PR \wedge p \rightarrow c \mid c \equiv a$$

5. Decentralized application architecture

Based on the system model elaborated in Section 4, we choose appropriate technologies and standards to implement a DApp for DT data sharing in Section 5.1. We implement our entities by leveraging several blockchain design patterns (Section 5.2). To showcase the inner workings of the DApp, the most important data flows for DT management are described in Section 5.3. The concomitant access control implementation is detailed in Section 5.4.

5.1. Technology selection

For the DApp prototype we rely on the Ethereum blockchain, which is commonly used for research, e.g. in blockchain-based business process management Haarmann, Batoulis, Nikaj, and Weske (2018). It offers the Turing complete smart contract programming language Solidity and has a large developer community, resulting in advanced development tools and vulnerability scanners Ayman, Aziz, Alipour, and Laszka (2019).

Fig. 4 depicts the technical architecture of the EtherTwin DApp. A **User Interface** simplifies the interactions of the DT lifecycle participants, such as creating twins and uploading data. For trustless interaction with the blockchain it is implemented using the single page application JavaScript framework Vue.JS³ – a server is only needed to serve static assets. The module ethereumjs-wallet is used for managing the user's blockchain account, providing access to the user's public and private key. Key pairs are dynamically created on first access and stored in the browser's local storage for future visits.

Web3.JS is used to send transactions signed with the private key to the **Smart Contracts** on the Ethereum blockchain. The front end is connected to an Ethereum blockchain node through a WebSocket connection. WebSockets improve performance over HTTP connections by providing a two-way communication channel between the client and the Ethereum node. This avoids the need to set up individual HTTP connections for each request Fette and Melnikov (2011). WebSockets also enable subscription to smart contract events (publish-subscribe style), which is utilized by the Device Agent for synchronization purposes. During development, we observed a significant speed up in page load times after switching to an RPC connection based on WebSockets.

The erebos module⁴ reuses the blockchain account to upload data to the **Off-chain Storage** based on the Swarm DHT. While Swarm is mostly known for its permissionless test network, it can also be deployed as a private DHT with a fixed set of peers. Swarm reuses Ethereum accounts as its identity system, which simplifies its integration as off-chain storage. Additionally, the data types used in both systems are compatible: References to Swarm data are encoded as 32 byte SHA3 hashes, which can be stored in a Solidity bytes32 variable in the smart contract. For dynamic content, Swarm provides Feeds. Feeds have a fixed address specified by user (Ethereum account) and topic (any SHA3 hash). They can only be updated by their owner with a public-key signature. Any Swarm user can read-access the most current and past updates. This concept is useful for sharing file keys and real-time sensor data under a fixed address, despite Swarm's content-addressed storage. Ethereum Swarm Contributors (2019)

³ vuejs.org

⁴ erebos.js.org

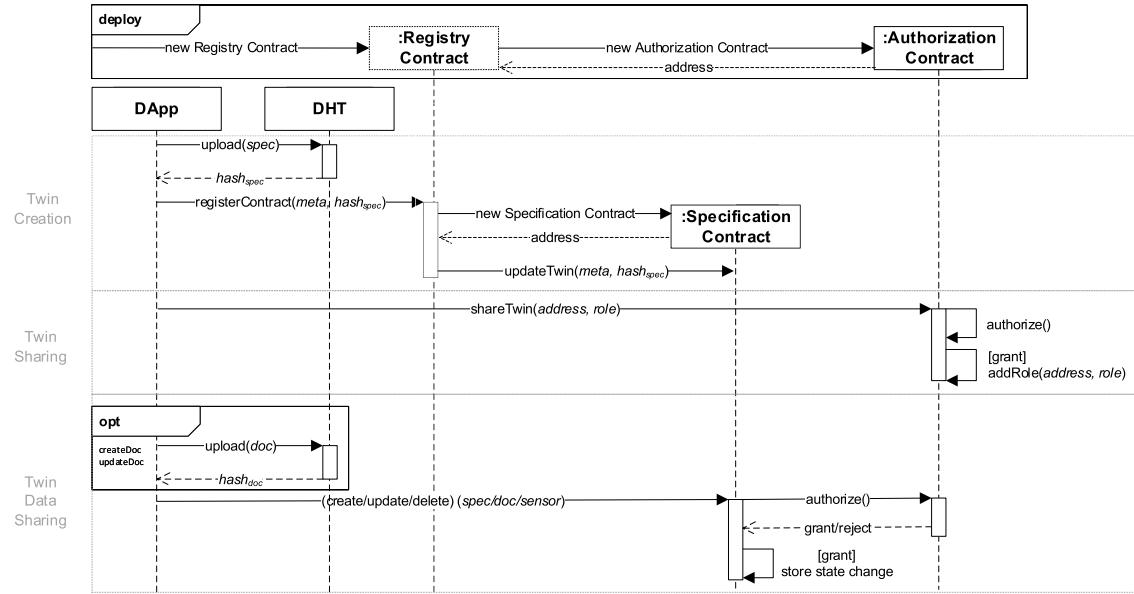


Fig. 5. Sequence diagram showing initial deployment and user interactions with the smart contracts and DHT.

The **Device Agent** bidirectionally synchronizes the DT's underlying Asset with the decentralized DT on Ethereum and Swarm. It runs as a node.js⁵ background process and monitors new sensor data from the asset, which is then uploaded to Swarm.

Like other authors creating DTs Eckhart and Ekelhart (2018); Schroeder, Steinmetz, Pereira, and Espindola (2016), our work relies on Automation Markup Language (AML), which is defined in the industrial standard IEC 62714. AML describes the specification of the asset including its components and their logic. Components are derived by parsing the AML-based asset specification.

5.2. Design patterns

Several blockchain application design patterns Xu, Pautasso, Zhu, Lu, and Weber (2018) are used in our prototype to address the requirements of DT data sharing. A *Contract Registry* pattern keeps track of individual DT contracts. A *Factory Contract* pattern is used to instantiate individual DT sharing instances. The access control model from Section 4.3 is implemented using the *Embedded Permission* pattern and implemented in the separate Authorization contract. The *Multiple Authorization* pattern is used to ensure that all sharing parties agree before changes to a DT contract are made. The *Off-chain Data Storage* pattern is used to meet the data volume and latency requirements. The Device Agent implements the *Reverse Oracle* pattern to mediate between the industrial asset and the distributed ledger. It monitors events occurring on the asset and publishes sensor data for authorized parties. Additionally, the agent is responsible for managing and distributing the symmetric file keys used for encrypting off-chain data, as detailed in 5.4.

5.3. Data flow

Fig. 5 shows how the contracts interact during the deployment, twin creation and sharing phases of the DT lifecycle.

Deployment. Initially, the Registry and Authorization contracts are deployed by the blockchain consortium initiator. The Specification contract template is deployed, but not yet instantiated as it is twin-specific.

Registration. When a user first opens the app, a new Ethereum account is created, represented by an Ethereum public-private key pair. The public key is shared off-chain by publishing it on the account's Swarm Feed. This avoids on-chain storage costs and allows anyone to retrieve the public key from the corresponding Swarm Feed. To improve usability and to avoid the need to share addresses out-of-band, we also register a mapping of the user's Ethereum address to a username on the Authorization contract.

Twin Creation and Sharing. On twin creation, the Owner provides a specification, which is parsed to extract the twin's components. A transaction is sent to the Registry Contract, which creates a new Specification Contract instance based on the provided data. In the authorization contract, the access control attributes of the newly created twin are initialized with the provided components. The AML-formatted specification is stored on the DHT and included with a hash reference. After a twin has been created, the Owner may share it by adding a role to the lifecycle participant's blockchain account.

Twin Data Sharing. Each transaction intending to create, update or delete an entity of the twin must first be authorized through the Authorization contract. It should be noted that deletion only removes the entry from the current state; the state's history is

⁵ nodejs.org

Table 3

Role mapping for entity Create/Read/Update/Delete permissions. \sim : Permission depends on presence of component attribute.

Permission	Twin				Document				Sensor			
	C	R	U	D	C	R	U	D	C	R	U	D
Device	x	✓	x	x	x	✓	x	x	x	✓	✓	x
Owner	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Manufacturer	x	✓	x	x	~	~	~	x	x	~	x	x
Maintainer	x	✓	x	x	~	~	~	x	~	~	~	x
Distributor	x	✓	x	x	~	~	~	x	x	x	x	x

preserved on the blockchain. If the action is authorized, the corresponding state change is registered in the smart contract state. For documents and specification version updates, the same procedure applies, except that metadata such as the filename remains unchanged.

Sensor Feed Updates. Due to latency requirements and to reduce the number of costly smart contract transactions, sensor data is shared off-chain. After a sensor is registered on-chain, the Device Agent connects to the corresponding component sensor and subscribes to its sensor data. Each new sensor entry is encrypted with the component's sensor encryption key and published to the sensor's Swarm Feed.

5.4. Access control implementation

In the following the decentralized implementation of the formal access control model detailed in [Section 4.3](#) is described.

Authentication. Authentication is based on blockchain accounts, which consist of a private key and an address. Identities are represented by addresses, which are created by hashing the public key. They are used for signing transactions and sharing confidential data intended for specific participants.

Authorization. Data stored on-chain is implicitly accessible to all participants storing the blockchain. For this reason, only metadata and off-chain references are stored in smart contract state. State change transactions require authorization by the Authorization contract authorization, with component-based entities (documents, sensors) also requiring the corresponding component attribute. The append-only nature of the blockchain ensures traceability of all changes.

The default mapping of permissions to roles is shown in [Table 3](#). Permissions comprise the CRUD operations for each of the main sharing objects *Twin*, *Document* and *Sensor*. The entities *External Source* and *Specification* do not have separate permissions and instead inherit the *Twin* permissions. Role and attribute mappings are controlled by the DT Owner and can be modified for each individual DT. For example, permissions may be removed from a role or attributes added to a user. These permissions are enforced on-chain by the Authorization contract. Read permissions for off-chain data are enforced by encrypting all data related to off-chain entities (*Specification Version*, *Sensor Entry* and *Document Version*). The encryption key is shared only with authorized users.

Encryption. All data is AES-256-encrypted before being uploaded to the Swarm DHT. Permissions are enforced by sharing a public-key encrypted version of the symmetric file key. Since Ethereum uses public keys based on elliptic curve cryptography, we rely on the Elliptic Curve Integrated Encryption Scheme (ECIES). However, Ethereum addresses are hashes of the public key and not the public key itself, which means they cannot be used for encryption. Therefore, participants additionally share their public key on their personal Swarm Feed (identified by their account address).

The file keys are then distributed on a Swarm Feed, which allows dynamic off-chain updates when new users gain permission. For the specification file, the asset Owner manages the file keys. For component-based entities, the file keys are managed by the Device Agent. The Device Agent must be trusted, since it has full access to the asset. It is thus able to enforce on-chain permissions for off-chain data continuously.

The Device Agent creates two unique symmetric keys for each component (for documents and sensors). File key recipients are determined based on roles and attributes stored on-chain. The corresponding algorithm for creating file keys is shown in [Algorithm 1](#). The formal notation is based on [Section 4.3](#). The algorithm must be executed for each twin before any files can be uploaded, since it distributes the symmetric keys needed for encryption. For this reason the Device Agent continuously monitors the blockchain for newly created *twins* managed by its address and associated permission updates. This is achieved by subscribing to contract events emitted by the *Authorization* contract. The Device Agent also subscribes to attribute and role change events. On each event, on-chain permissions are retrieved and the corresponding file keys are added/removed accordingly.

6. Evaluation

To evaluate the proposed DApp architecture, we follow a methodological approach based on Venable et al.'s framework for evaluation in Design Science Research [Venable, Pries-Heje, and Baskerville \(2012\)](#). The goal is to ensure both rigor and efficiency of our research. In our ex-post evaluation, we utilize both artificial (prototype, technical simulation) and naturalistic (case study, expert interviews) evaluation methods.

The screenshot shows the Ether Twin home menu. At the top, there is a navigation bar with the Ether Twin logo, a search icon, and links for Twins, Specification, Components, Documents, Sensors, and Data sources. A user profile icon is also present. Below the navigation bar, the title "Digital Twin Overview" is displayed. A sub-header says "Add a new twin or select a twin by clicking on an action." To the right are two buttons: "Add Twin" and "Refresh". The main area is a table with four columns: Name, Contract Address, Role, and Actions. The table lists three twins: "Boring and Milling Machine Twin" (Owner), "Candy Factory Twin" (Owner), and "Chemical Plant Twin" (Maintainer). Each row has a set of icons for search, add, edit, delete, and refresh operations.

Fig. 6. Screenshots of the prototype's home menu.

The screenshot shows the Ether Twin interface for managing components of a twin. On the left, there is a sidebar with sections for "Sensor Feed: Machine logs" (listing timestamped log entries for machine programs and states), "Documents for Twin: Boring and Milling Machine Twin" (with a form to add new documents), and "Existing Documents" (listing a manual for Siemens HMI panels). On the right, the main content area is titled "Components of Twin: Boring and Milling Machine Twin". It shows a hierarchical tree structure with nodes for "PhysicalNetwork", "LogicalNetwork", "HMI", "Switch1", and "PLC1". Each node has a small icon and a detailed description below it. For example, the "HMI" node shows the ID "ID : 5774a072-dae4-4bbe-9db7-8fe0ac3938f4" and a list of elements: "Portlist (28dad2fb-8949-44fb-beff-aa1bac51beb1)" and "HMI (068ec45a-1002-4a75-8e27-21d8e0da6e3d)".

Fig. 7. Screenshots of the prototype's component-based structure and information management.

We first describe the EtherTwin prototype and its user interface in Section 6.1. The prototype is evaluated with several technical experiments concerning latency and cost in Section 6.1. Its practical application is explained via an industry use case in Section 6.3. Finally, we interview several industry experts regarding the prototype's benefits and remaining challenges in Section 6.4.

6.1. Prototype

The EtherTwin prototype is available on GitHub⁶, including a video demonstrating the use case illustrated in Section 6.1. It consists of about 3000 single lines of code (SLOC) for the DApp and Device Agent, as well as 400 SLOC for the smart contracts. We analyzed all smart contracts for vulnerabilities using the SmartCheck vulnerability scanner Tikhomirov et al. (2018). Hereafter, screenshots are presented to show the prototype's functionality.

The prototype's start page is illustrated in Fig. 6. It gives an overview of the twins the user is involved with, and shows the role of the user for each twin. The navigation bar shows the available pages for the selected twin that is highlighted in gray. Navigation to the respective pages is handled by clicking on the respective icon in the twin's row. The icon shown on the very right of the navigation bar provides a visual representation of the user's network address. It leads to the account page, containing information about the network and current user.

Fig. 7 contains three screenshots that show the component-based organization of the prototype per twin. The screenshot on the

⁶ <https://github.com/sigma67/ethertwin>

Fig. 8. Screenshots of implemented access control mechanisms.

right shows the composition of the selected asset/twin with its components and sub-components. This structure is parsed from the AML specification, which is required for twin creation. The upper left screenshot illustrates the sensor feed capabilities. The screenshot on the lower left shows the existing documents per twin. Each document is thereby assigned to a component. For each component, documents from each lifecycle-phases can be uploaded. However, users can only download, upload or update a document to a component if they have the respective component attribute in the smart contract. In practice, each user should be assigned the component attributes that the user is involved with in the lifecycle.

Fig. 8 shows the prototype's role and attribute management page. In the EtherTwin prototype, the Owner of a twin can see all other involved users and their lifecycle involvement. Furthermore, the Owner can handle the access to the resources as shown in the screenshots below. The screenshot on the bottom left side shows how the user's role can be changed, while the screenshot on the right side illustrates the adjustment of the user attributes.

Further screenshots of the prototype can be found in [Appendix A](#) ([Figure A1](#), [Figure A2](#), [Figure A3](#)) and in our GitHub-repository⁷.

6.2. Technical experiments

To evaluate the performance of our prototype, we first consider latency of the interactions described in the prototype. Our prototype environment is set up on a Raspberry Pi using Parity Ethereum 2.7.2 and Swarm 0.5.7. The DApp and Device Agent were run on an i7-8550U CPU.

When a new twin is created, the Device Agent must create the twin's symmetric encryption keys before any data can be shared. To evaluate this latency, we benchmarked the runtime of [Algorithm 1](#). The algorithm runs every time a DT is created or its permissions are updated. It only runs once the transaction is included in the blockchain, since it is triggered by smart contract events. The results in [Fig. 9](#) show that the runtime is on the order of one to three seconds. This is sufficient for real-world scenarios, since sharing interactions are not immediate. The runtime is not significantly affected by the number of users the DT is shared with. It increases only slightly with the complexity of the asset specification (number of components).

To ensure user adoption, interactions with the user interface should have low latency. Each time a smart contract transaction is issued, the user needs to wait for a blockchain confirmation. Therefore, we measured the latency of interactions with private and public blockchains in [Table 4](#).

Another aspect relevant for public blockchain deployments are transaction costs for the Ethereum smart contracts. The

⁷ <https://github.com/sigma67/ethertwin/tree/master/misc/Screenshots>

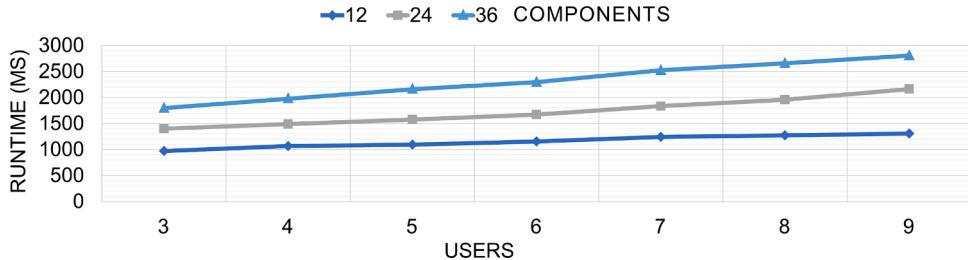


Fig. 9. Runtime values for Algorithm 1 for varying numbers of users and components.

Table 4

Latency (ms) and cost (ETH, €) for contract deployment and interactions. Gas price: 10 Gwei, 120 € /ETH.

Action	ms	Gas	ETH	€
Initial Deployment	-	14,548k	0.14548	17.46
Twin Creation	896	4576k	00.4576	5.49
Twin Sharing	353	144k	00.0144	0,17
Specification Version Creation	262	94k	00.0094	0,18
Document Creation	485	254k	00.0254	0,50
Document Version Creation	365	99k	00.0099	0,19
Sensor Creation	374	95k	00.0095	0,18
Attribute Update	276	50k	00.0050	0,10

cryptocurrency costs are shown in Table 4. Sub-second latencies demonstrate that the user experience is fluid, despite client-side encryption/decryption and network delays. Costs are also quite low except for *Twin Creation*, since a new contract is instantiated. The use case costs include each action once, except for *Document Creation*, since two documents are created. In practice, lifecycle participants should decide based on usage cost projections to either jointly run a private network with operational costs, or use the public Ethereum network with the associated transaction fees.

6.3. Use case

This use case illustrates how our EtherTwin DApp is used in practice by creating a DT based on real enterprise data that is provided during the Secure Industrial Semantic Sensor Cloud (SISSeC) project⁸. The SISSeC project focuses on introducing Industry 4.0 in small and medium enterprises (SME) and aims at securely unifying and analyzing machine data. The industrial assets targeted in the project are part of the manufacturing process of printed circuit board (PCB) panel prototypes of a small German enterprise. The central goal for the PCB panel manufacturer is to gather all data available about the machines, to unite and analyze the data. Thereby, novel insights such as the determination of flaws in the manufacturing process present the desirable outcome.

Our DApp prototype unifies the available data of an industrial asset throughout all lifecycle phases. In this use case, we create a DT for the boring and milling machine that gouges holes into the PCB panels. The demonstration of the implemented use case can be found online⁹ and its manifestation can be gathered from the screenshots of the prototype (Section 6.1 and Appendix A).

At first, the machine specification in the form of an AML-file is implemented to set up the respective smart contracts for the use case (cf. Fig. 2). Then the feed data from the machine is integrated from sensors, ranging from sensors determining the position of the drill to logs of the PLCs concerning the running program. Moreover, we unified asset-relevant documents like manuals of the machines' ICSs¹⁰.

Thereby, the documents are assigned to their corresponding component. For instance, a manual of a Siemens S5 PLC is assigned to the PLCs of the boring and milling machine. Currently, we created user accounts for the PLC's manufacturer, the machine operator and the maintainer of the machine's motors for demonstration. However, there are other users that can be included, e.g. the manufacturer of the motors, the maintainer of the PLCs and HMI or the distributor of the barcode reader.

Based on an interview with the CEO and the CIO of the firm that currently operates the boring and milling machine, we gather that our EtherTwin prototype meets their current needs for central collection of data about their machine. For example, when service is required, the operator usually has trouble providing the right information to the maintainer. However, this information is needed for the maintenance service to bring the right tools and rapidly assesses the machine's state and problem. In their view, EtherTwin poses a solution to this issue. Moreover, they consider the component-based data management a useful strategy that facilitates their search of

⁸ <https://www.it-logistik-bayern.de/produktionslogistik/projekt-sissec>

⁹ <http://ethertwin.ur.de>. The use case can be tested with a demo Owner account with the private key 1bed7-c10358ece007522558c4801b84424750f5a626ce5c9093411c9fc197a6f, to be entered on the account page (top right icon)

¹⁰ Please note that the SISSeC project is at an early stage, where more data about the machine is still to be gathered.

information about sub parts. Nevertheless, the interviewees also state that EtherTwin's access control mechanisms are very valuable to prevent knowledge drain. Nevertheless, it is uncertain whether lifecycle participants have the required knowledge to install the proposed solution.

This use case shows that our DApp supports our goal of unifying asset-relevant data among its lifecycle with its participants. This results in enabling a feedback loop among the machine's lifecycle phases. The participants of the lifecycle phases can harness this information to optimize their own business.

6.4. Expert interviews

To validate our prototypical implementation of blockchain-based DT information management, we conducted semi-structured interviews with industry experts. The goal of the interviews is to determine the prototype's conformance to practical requirements and to identify potential adoption barriers.

Participants We conducted semi-structured interviews with ten industry experts from six different enterprises. The industrial domains the experts have experience with include engineering industries (4 experts), manufacturing (2 experts), logistics (1 expert) and IT firms and blockchain corporations (3 experts). Four of the investigated experts have a security background, while two of these are security information architects, whereby one is designing secure blockchain architectures. Another expert is responsible for security lifecycle and governance and the last one is tackling IoT security in particular. Two of the remaining experts work exclusively on blockchain technology and another works as an information architect. The last three experts are IT consultants.

The experts have a cumulative 101 years of experience, ranging from 2 to 25 years with an average value of 10.1 years and a median of 7.5 years. This experience was gained in companies of various sizes, including both SMEs (with up to 249 employees) and large enterprises (up to 500,000 employees). The average enterprise size the interviewees are familiar with is 164,583 with the median at 30,000 employees.

Procedure To identify the opportunities and challenges of using our blockchain-based DT data management approach and to evaluate the implemented prototype, we develop three categories of questions for the interview. These categories are based on DT **lifecycle aspects (1)**, the suitability of the **blockchain approach (2)** and the characteristics of the developed **prototype (3)**. The questions for the interview are based on relevant literature. We follow Dietz et al. (2019) and Dietz and Pernul (2020a) to identify DT lifecycle aspects (1). For (2), we rely on Malakuti and Grüner (2018) and Rubio et al. (2017) that provide the problem area to which our approach poses a solution. To derive the questions for category (3), we derive the questions from the distinct features of our prototype (cf. Table 1).

To evaluate and gain additional practical insights on the categories (1), (2) and (3), we conduct a semi-structured expert interview according to Lazar, Feng, and Hochheiser (2017). The interview is structured in the following phases:

- Phase 1) Introduction. At the start, the participants are questioned about their expertise and practical experience. Subsequently, an introduction to our research problem and approach is given. Additionally, we guide each interviewee through our EtherTwin prototype. Before the experts are interviewed, we encourage them to mention any issues that emerge during the following phases.
- Phase 2) Interview. In this phase, the set of questions corresponding to the three categories are posed. Thereby, the interview questions are deliberately stated in a generic way to enable experts to share their individual experience Lazar et al. (2017). The questions start off with **lifecycle aspects (1)**, which represent the most generic questions, followed by requesting the experts' opinion on the underlying **blockchain approach (2)**. The last category contains the least generic questions and tackles our EtherTwin **prototype (3)**.
- Phase 3) Wrap-up. We summarize the experts' main feedback. The expert is encouraged to state additional feedback on our research and EtherTwin prototype to help validate our approach. Moreover, areas requiring revision can be identified.

The guideline to the expert interview, including the interview questions, procedure and research purpose, can be found in Appendix B. Each interview participant received a copy of the guideline in advance of the interview.

Results We briefly describe the results of the interviews below, before discussing the experts' suggestions for improvement in Section 7.

In terms of relevant **lifecycle aspects (1)**, the experts mentioned that there are additional roles at each operator that need separate permissions, for example engineers, managers, developers, analysts and security employees. Half of the experts believe that including relevant participants such as auditors and regulatory authorities could be beneficial. Moreover, 30% of the experts think it would be beneficial to include roles for public authorities, e.g. to manage the compliance to environmental law. Two experts mention that the Owner and operator may not be the same. For instance, the operator might only have leased the industrial asset, while the Owner might still be the integrator or another lifecycle participant. Moreover, some of the roles should be further distinguished between manufacturers of the components, the integrator of the components (the manufacturer of the machine) and the operator. Another helpful remark, mentioned by two experts, is that modeling sub-roles might be required.

Six out of ten experts see the data for managing an industrial asset as dependent on various aspects, including the industrial asset itself, the lifecycle phases involved as well as the use case, as also other non-industrial devices could be modeled with our approach. However, the most important data was:

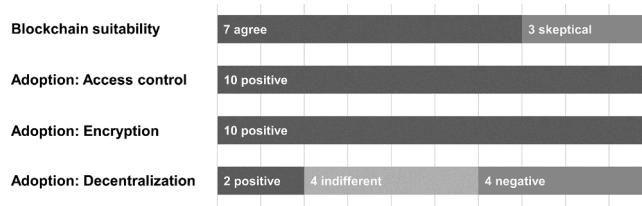


Fig. 10. Expert evaluations regarding the suitability of the presented solution.

- sensor and operating data (50%)
- cumulative information & analytics including dashboards (50%)
- master data categorized into mechanical, electrical and IT-related (30%)
- the relations among the components, e.g. dependencies, network (20%)
- as well as information about hardware- and software and their modifications (10%)

Three experts reckon the sensitivity of the data as a very important aspect, esp. when critical infrastructures and functions are involved.

Information about an industrial asset is currently shared, but only in a limited way. According to the experts, Industry 4.0 sharing practices are currently still in an early stage. For example, ICSs are integrated to corporate IT systems in order to communicate with enterprise resource planning (ERP) and e-commerce systems. Moreover, according to the experts, current sharing practices involve only strategic partners. Nevertheless, the creation of greater collaboration platforms is planned – leaning towards the notion of an ecosystem of DTs. In practice, our sharing approach is thus seen as a future issue, while security is and will remain a pressing challenge.

Thereby, the most important advantages of information sharing are collaboration opportunities and product improvement (70%), followed by transparency and recording (40%). Using the blockchain, the experts expect a very low rate of failure (availability) and manipulation (integrity). The greatest disadvantage are data ownership issues and the potential loss of valuable corporate trade secrets (70%). Four experts expect the risk of industrial espionage (esp. among the supply chain), resulting in an increase of the substitutability potential through rivals and in a reduction of lock-in effects. Nevertheless, two experts emphasized that the benefits certainly outweigh the downsides.

Regarding **blockchain suitability (2)**, the answers are summarized in Fig. 10. Most experts agreed that blockchain is suitable for managing DT data along its lifecycle. Nevertheless, some experts were skeptical and preferred a TTP over a blockchain solution for its simplicity. One expert noted that this solution should not be used for machine operations since that would require millisecond latencies. Another interviewee suggested that it should be used to track machine interactions, i.e. firmware upgrades and part changes.

Finally, the user interface of the EtherTwin **prototype (3)** was received positively. All participants agreed that it was well suited to the task at hand. Adjectives used for description were *intuitive*, *clear* and *modern*. One expert argued that the developed user interface is not needed in practice, since the backend should be fully integrated with existing systems, such as condition-based maintenance systems, ERP and product data management (PDM).

When asked for estimates of practical performance requirements, the experts provided varying estimations based on their experience. While an SME with 10 manufacturing machines may create two twins of these machines per year, an automotive manufacturer may create one per produced car, or as many as 10,000 per month. Estimates for shared documents for a twin also ranged from one document per day to a few documents per year, depending on the amount of shared documentation (i.e. aircraft production requires a large number of accompanying documentation material). For sensor data, raw sensor logs can result in significant data volume and velocity (up to terabytes/day), but not all of this data requires sharing. Experts suggested that only non-nominal or aggregated sensor data needs to be shared, resulting in a volume around hundreds of entries per hour.

7. Discussion

Hereafter we discuss the results of the evaluation, the resulting limitations and how the experts' feedback can be used to improve the prototype in the future. We start with discussing the lifecycle aspects in Section 7.1. At last, performance (Section 7.2) and security (Section 7.3) aspects are discussed.

7.1. Lifecycle

Access Control. Additional lifecycle roles (e.g. an auditor or government authorities) could be implemented by updating our Authorization Contract. This includes the possibility for sub-roles and inheritance, for example to separate permissions for a technician and financial controller at the manufacturer. Delegation of rights could be achieved by including permission delegations in ABAC, for which several strategies have been proposed [Servos and Osborn \(2016\)](#). Another suggestion concerned the need for time or

event-based access to data by lifecycle parties. The access control model could be expanded to include an expiry time for each attribute, which is validated whenever access is requested. EtherTwin provides a starting point that can be extended and specialized to fit practical use cases individually. Another recurring suggestion made by experts was a role-specific user interface. In addition to tailoring the available roles to the practical use case, a role-specific twin overview page could help users find the needed information faster.

Data Governance. For collaboratively run applications, governance aspects are important to consider. Future software updates to the deployed smart contracts may be necessary to incorporate additional DT features. Code changes to deployed smart contracts are not trivial and require specific application patterns to avoid data loss. Upgradability of smart contracts can be achieved using a Contract Registry or a Data Contract pattern Xu et al. (2018). To create new twins with enhanced functionality, the existing Registry contract can be upgraded to allow for modular Specification contract templates.

Data structure. Additionally, a data structure might be established that is not only based on the components (cf. Fig. 3), but categorizes mechanical, electrical and IT information as well. Future work could investigate how to integrate this categorization, e.g. as an alternative structure or as sub-structure for each component.

Additional Data. Our prototype has few restrictions regarding data volume and variety. The additional information deemed relevant by the experts could thus be easily integrated. Additionally, simulation is a key part of DTs. EtherTwin currently supports upload of simulation data, but future work could investigate how simulations can be directly deployed in the user interface. For better usability, future work could also extend our prototype by including analytical dashboards. Experts suggested that each role should be able to get an at-a-glance overview of the asset's state. Such a dashboard could include out-of-range sensor values, recently updated documents, asset performance metrics and risk indicators.

Asset Control. Similarly, DTs should provide some control over the industrial asset. Firmware management and updates were suggested by experts as a potential use case for EtherTwin. Program code of PLCs could be uploaded through the user interface by permissioned users and automatically installed by the Device Agent, documenting all actions in the smart contract. This enables traceability and accountability of participants for each modification made to the physical asset.

7.2. Performance

On-chain. The twin and document creation rates estimated by the experts do not present a challenge for a prototype, as even the maximum values are within the performance limits of Ethereum and Swarm. Private Ethereum blockchains support between 50 and 100 transactions/second Dinh et al. (2017), which implies that more than 4 million twin interactions are possible per day (i.e. document creation, sensor creation).

Off-chain. Experts mentioned that multiple events might need to be shared per second for a specific sensor. However, Swarm is currently restricted to one update per feed per second. To deal with this restriction, sensor feeds are updated once per second with batched sensor updates from the Device Agent. Thus, no data is lost and failure data is shared in a timely fashion. This restriction precludes real-time monitoring and control of assets, as pointed out by an expert.

7.3. Security

Research Question. With our research question we aim to develop secure lifecycle information management for DTs:

RQ1. How can the data of Digital Twins be shared among multiple untrusted lifecycle parties while ensuring confidentiality, integrity and availability?

Our prototype provides *confidentiality* by including fine-grained access control as well as encryption. All experts agreed that access control and the concomitant encryption are essential for business adoption (cf. Fig. 10). On-chain data *integrity* is assured by the immutability of the blockchain and the full replication of data among the participating nodes. Off-chain data integrity is provided by maintaining the DHT encryption key and sensor feeds through the Device Agent controlled by the Owner. Additionally, Swarm feed updates must be signed and are append-only, so integrity of past entries is maintained. In terms of *availability*, our decentralized approach enables the participants to manage their own nodes to maintain fully replicated copies of on-chain and off-chain storage. The proposed architecture relies on three distinct systems to function properly: the blockchain network, the DHT and at least one Device Agent per organization. Due to the resulting complexity, consequences of failure should be properly considered:

- **Blockchain node failure:** If an organization's blockchain node crashes, it will be unable to access the DApp as it relies on the blockchain node as a data source. This would not affect other organizations. If $> \frac{1}{3}$ of all blockchain nodes in the network fail, write transactions are no longer available for all participants
- **DHT node failure:** If the DHT node is unavailable, the organization will be unable to retrieve the DT specification, documents and sensor data. Other organizations are unaffected, unless they are trying to retrieve Twin data of the crashed organization for the first time

- **Device Agent failure:** A failed Device Agent implies that encryption keys will not be updated on permission changes while it is down. Thus, newly shared data for its twins will not be available to the sharing recipients. Additionally, sensor data will not be updated.

Encryption. Despite these already built-in security measures, sharing business data with external entities bears risks for enterprise security. While data is encrypted, loss of the encryption key or compromise of the elliptic curve/AES encryption schemes could lead to access by unauthorized parties. Since data cannot be removed from other nodes once uploaded to the DHT, there is an inevitable loss of control that comes with sharing encrypted data. Due to the distributed nature of the DHT, read access to shared data cannot be revoked, and it is not possible determine which users actually accessed DHT data. Safeguarding the Device Agent and the encryption keys are thus paramount to data security in our approach. The prototype could be improved in this regard by hiding private key information in the user interface and using the Web Cryptography API¹¹ instead of the browser's local storage. In addition, the sharing enterprise must rely on the recipients to protect the encryption keys and data as well. Future research could also investigate future-proofing the encryption procedures by utilizing quantum-proof schemes.

Misuse. Another consideration mentioned by an expert is misuse potential at the time of data entry. For a decentralized application, besides signature checks there is no way of checking the authenticity of uploaded data. A malicious lifecycle participant could thus upload false information that cannot be deleted. Nevertheless, the versioning system in EtherTwin ensures that past versions of data remain available.

Public blockchains. If a public blockchain is used, confidentiality of on-chain metadata becomes a concern. Since on-chain data is not encrypted, participants should avoid including confidential information in metadata. If this is maintained, the contracts can be deployed on the public Ethereum blockchain and there is no need for participants to operate a blockchain infrastructure. To ensure infrastructure control and data confidentiality, both the Ethereum blockchain and the Swarm DHT can also be set up as private networks. Permissioned Ethereum networks may use a more resource-efficient byzantine-fault tolerant consensus algorithm such as IBFT 2.0 [Saltini and Hyland-Wood \(2019\)](#).

Identity Management. Usability could be improved by mapping Ethereum addresses to human-readable names. Organizations may associate employee identities in existing identity management systems with Ethereum key pairs to enable single sign-on. Future research could investigate how to best implement a mapping of enterprise identity to blockchain identity.

8. Conclusion

To conclude, the EtherTwin DApp implements the complex DT sharing requirements of the Industry 4.0 landscape without the need for a TTP. This is achieved through a fine-grained blockchain-based access control model coupled with encrypted off-chain data storage. The open source prototypical implementation is based on Ethereum and Swarm. Additionally, we evaluate our model through use case elaboration and performance testing. Interview responses by industry experts validate the prototype's practical suitability and provide avenues for future research.

For example, our work on blockchain-based information sharing and access control may be extended to other areas, i.e. health DT data sharing, data marketplaces and machine certifications. Business processes can also be interpreted as DTs [Dietz and Pernul \(2020a\)](#). Approaches for blockchain-based business process management involving physical assets could thus be integrated with blockchain-based DTs modeled in our work. Additionally, our prototype could be enhanced by enabling data flow from the twin to the industrial asset. These interactions could involve calling PLC functions through the smart contract, or installing firmware updates. Finally, simulation environments could be directly integrated in the decentralized sharing platform, instead of only sharing simulation results as documents.

Declaration of Competing Interest

The authors declare that they do not have any financial or nonfinancial conflict of interests

Acknowledgements

We would like to thank the interviewed experts for their time and valuable contributions. Furthermore, we would like to express our thanks to our reviewers for their helpful suggestions. Part of this work was performed under the ZIM SISSeC project¹², which is supported under contract by the German Federal Ministry for Economic Affairs and Energy (16KN085725).

¹¹ <https://www.w3.org/TR/WebCryptoAPI/>

¹² <https://www.it-logistik-bayern.de/produktionslogistik/projekt-sissec>

Appendix A. Screenshots of the prototype

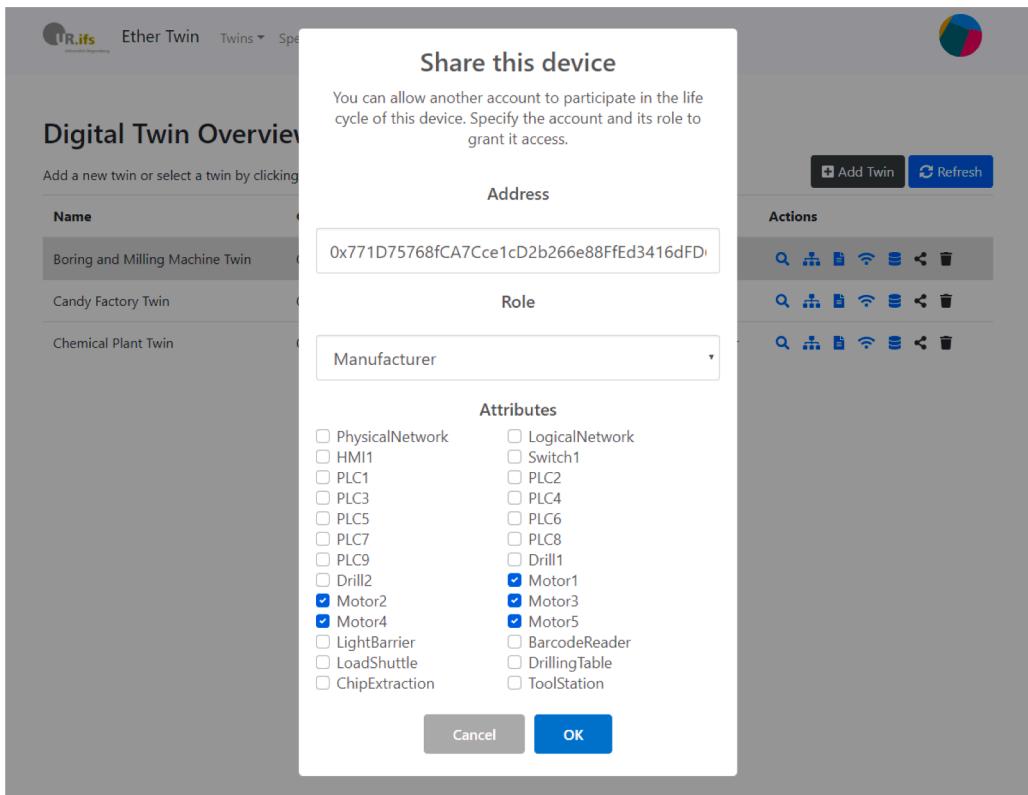


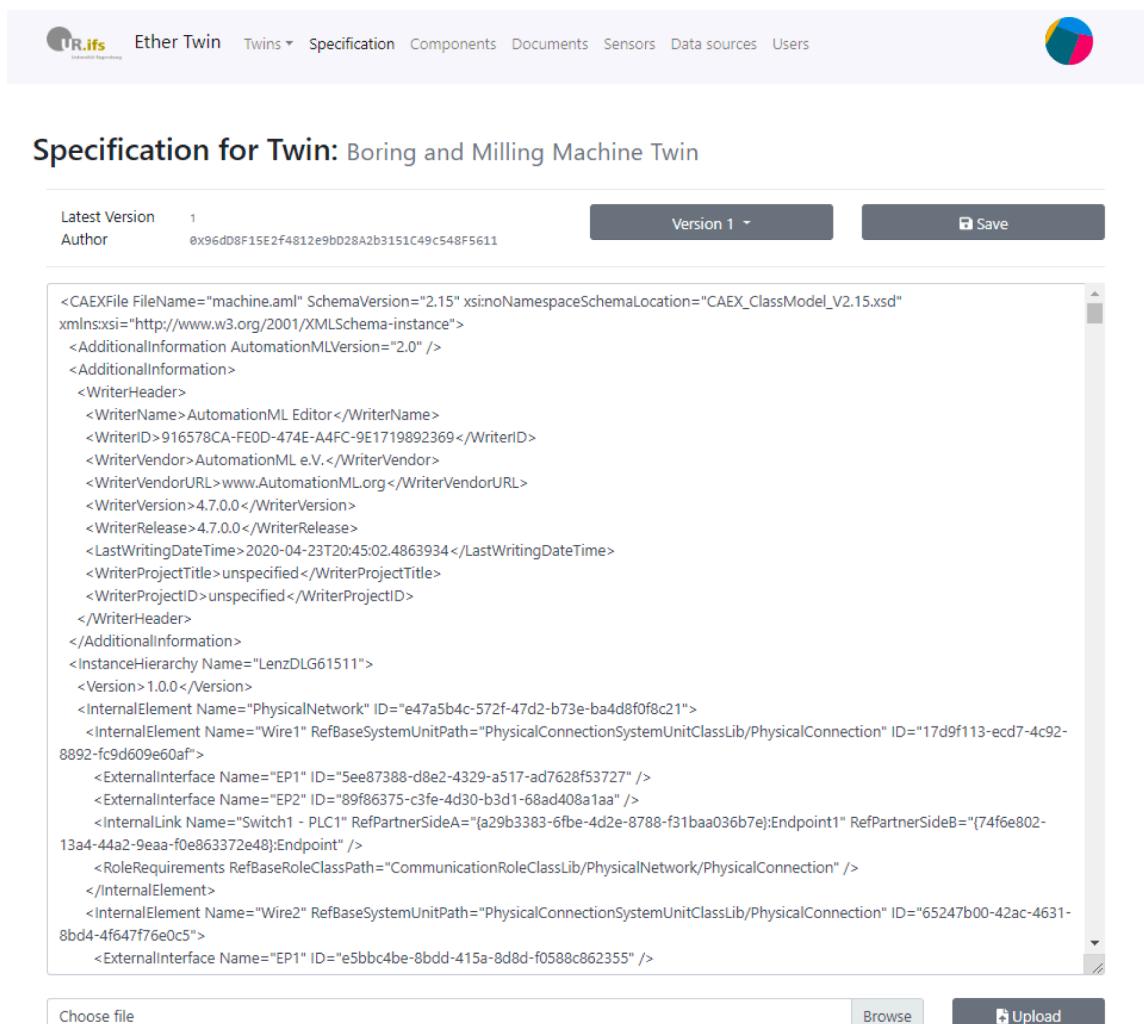
Fig. A1. Screenshot of the "share twin"-functionality.

The screenshot shows the "Account Information" page. It displays the following details:

- Network:** private
 - Registry Contract: 0xC2D067cdC767E1702Ba646b4C1cD619Bb7a6AD37
 - Authorization Contract: 0x45C10F7f3f4f3779aE595Ec69e2b5561C80Adc8f
- Account:** 0x96dd8f15e2f4812e9bd28a2b3151c49c548f5611
- Balance:** 1345.68149378 ETH
- Public key:** 0092f76b4aef930b22fb01adcb8d0089a0d53dcee3a4750fdc12cb191f0c605303a3a3e493752a22c256922801ee07be45780369afee4a84f4675f0884df5bf8
- Private key:** 1bed7c10358ece007522558c4801b84424750f5a626ce5c9093411c9fc197a6f

Below this, there is a section for "Update private key" with a text input field and a "save changes" button.

Fig. A2. Screenshot of the user's account page.



The screenshot shows the Ether Twin software interface. At the top, there is a navigation bar with the logo of IR.IFS (University Regensburg), the title "Ether Twin", and links for "Twins", "Specification", "Components", "Documents", "Sensors", "Data sources", and "Users". On the right side of the header is a blue circular icon.

Specification for Twin: Boring and Milling Machine Twin

Latest Version 1 Author 0x96dD8F15E2f4812e9b028A2b3151c49c548F5611

Version 1 Save

```
<CAEXFile FileName="machine.aml" SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <AdditionalInformation AutomationMLVersion="2.0" />
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>AutomationML Editor</WriterName>
      <WriterID>916578CA-FE0D-474E-A4FC-9E1719992369</WriterID>
      <WriterVendor>AutomationML e.V.</WriterVendor>
      <WriterVendorURL>www.AutomationML.org</WriterVendorURL>
      <WriterVersion>4.7.0.0</WriterVersion>
      <WriterRelease>4.7.0.0</WriterRelease>
      <LastWritingDateTime>2020-04-23T20:45:02.4863934</LastWritingDateTime>
      <WriterProjectTitle>unspecified</WriterProjectTitle>
      <WriterProjectID>unspecified</WriterProjectID>
    </WriterHeader>
  </AdditionalInformation>
  <InstanceHierarchy Name="LenzDLG61511">
    <Version>1.0.0</Version>
    <InternalElement Name="PhysicalNetwork" ID="e47a5b4c-572f-47d2-b73e-ba4d8f0f8c21">
      <InternalElement Name="Wire1" RefBaseSystemUnitPath="PhysicalConnectionSystemUnitClassLib/PhysicalConnection" ID="17d9f113-ecd7-4c92-8892-fc0d609e60af">
        <ExternalInterface Name="EP1" ID="5ee87388-d8e2-4329-a517-ad7628f53727" />
        <ExternalInterface Name="EP2" ID="89f86375-c3fe-4d30-b3d1-68ad408a1aa" />
        <InternalLink Name="Switch1 - PLC1" RefPartnerSideA="(a29b3383-6fbe-4d2e-8788-f31baa036b7e);Endpoint1" RefPartnerSideB="{74f6e802-13a4-44a2-9eaa-f0e863372e48};Endpoint" />
        <RoleRequirements RefBaseRoleClassPath="CommunicationRoleClassLib/PhysicalNetwork/PhysicalConnection" />
      </InternalElement>
      <InternalElement Name="Wire2" RefBaseSystemUnitPath="PhysicalConnectionSystemUnitClassLib/PhysicalConnection" ID="65247b00-42ac-4631-8bd4-4f647f76e0c5">
        <ExternalInterface Name="EP1" ID="e5bbc4be-8bdd-415a-8d8d-f0588c862355" />
      </InternalElement>
    </InternalElement>
  </InstanceHierarchy>

```

Choose file Browse Upload

Fig. A3. Screenshot of the AML-structured specification of the asset.

Data: A set of twins $T = (t_1, \dots, t_n)$ with mappings for roles M_t^{UR} , permissions M_t^{PR} , attributes M_t^{UA} and a set of components C_t .
Result: A set of encrypted file keys $fk_{tcu} \forall t \in T, c \in C, u \in U_t$ used to decrypt data $D_{tcn} \forall n \in 1..N$ and uploaded to DHT feeds owned by the Device Agent.

```

1: function CREATEFILEKEYS( $t$ )
2:    $C_t \leftarrow getComponents(t)$                                      ▷ retrieve permissions from smart contract
3:    $M_t^{UR} \leftarrow getRoleAssignment(t)$ 
4:    $M_t^{PR} \leftarrow getPermissionAssignment(t)$ 
5:    $M_t^{UA} \leftarrow getAttributeAssignment(t)$ 
6:   for each  $c \in C_t$  do                                         ▷ generate two symmetric keys  $sk$  per component
7:      $(sk_{tc}^{doc}, sk_{tc}^{sensor}) = \mathcal{G}_{enc}$ 
8:   end for each
9:   for each  $u \in (M_t^{UR} \cap M_t^{UA})$  do                               ▷ prepare file keys  $fk$  for users
10:     $pk_u \leftarrow (\text{DHT})\ getUserPublicKey(u)$ 
11:    for each  $c \in C_t$  do
12:       $r \leftarrow M_{tu}^{UR}$ 
13:      for each  $d \in \{doc, sensor\}$  do
14:        if  $(c \equiv a \mid a \in M_{tu}^{UA}) \wedge (p_{read}^d \in M_{tr}^{PR})$  then
15:           $fk_{tcdu}^d = ECIES\_enc(pk_u, sk_{tc}^d)$ 
16:           $(\text{DHT})\ updateFeed(c, fk_{tcu}^d)$ 
17:        end if
18:      end for each
19:    end for each
20:  end for each
21: end function
```

Algorithm 1. Create off-chain file keys based on read permissions.

Appendix B. Guideline of the expert interview

The present research focuses on blockchain-based Digital Twin information management (“*EtherTwin: Blockchain-based Digital Twin Information Management*”). It aims at developing an approach to securely share and store data about an Industry 4.0 asset (Digital Twin data) along its entire lifecycle. As the involved lifecycle parties that share data usually pursue different goals, trust issues hamper data sharing in practice. To resolve these trust issues and to strengthen security, blockchain technology is harnessed in the form of a Decentralized Application (DApp).

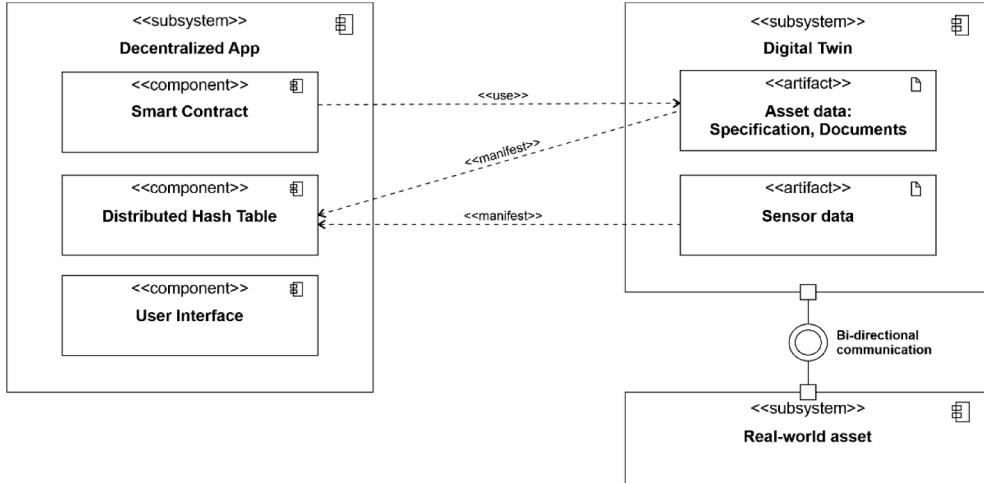


Figure 1: The concept of DT data sharing and storage realized with a Decentralized App (DApp), with its basis on the blockchain technology.

The *Digital Twin* describes a concept to virtually represent an Industry 4.0 asset (e.g. a conveyor belt). Thereto, a central collection of data of the industrial asset is of utmost importance. In **Figure 1**, the core data artifacts of a Digital Twin as well as its connection to the *Real-world asset* the Digital Twin represents are shown on the right-hand side. Moreover, the connectors show where each data artifact of the *Digital Twin* is represented in the *DApp*-architecture. First, the information of the asset’s *Specification* is used to create a *Smart Contract*. For instance, this includes data about the asset’s composition (*Specification* data). Asset data like *Documents* and *Sensor data* are stored in the *Distributed Hash Table* off-chain and all interactions with these data (upload, delete, update etc.) are linked in the blockchain. A *User Interface* builds on top and enables a user-friendly interaction from all lifecycle parties, including less technically savvy users.

The goal of this research is a full-featured implementation of Digital Twin data management, the EtherTwin prototype, and its evaluation. To enable secure data sharing, our approach ensures that integrity, confidentiality and availability of data are maintained.

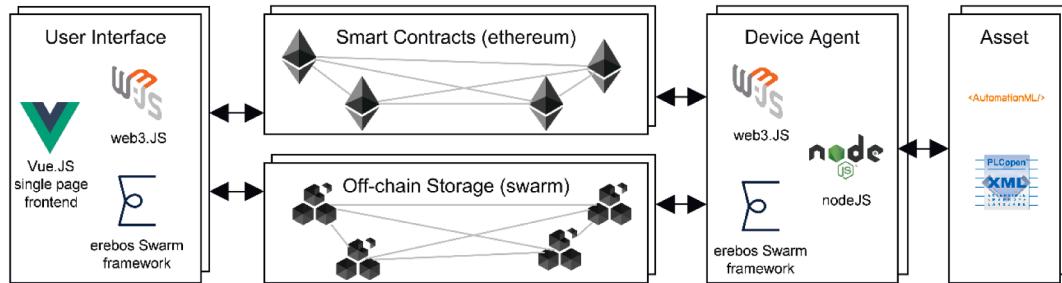


Figure 2: The selected technologies for the EtherTwin prototype.

Figure 2 shows the technologies used to build the EtherTwin prototype. The asset's *Specification* is based on the industry format [AutomationML](#). With this information, the components (industrial control systems) of the asset can be derived and a Digital Twin management space is built. The blockchain part is realized with Ethereum. An off-chain *Distributed Hash Table* is integrated. The Device Agent manages the connection to the sensors etc. of the industrial assets and directly incorporates sensor data and system log data.

Next to implementing the approach in form of the EtherTwin prototype, we conducted performance tests and proposed a use case from practice that shows how our prototype can be used. In addition, expert interviews are to be conducted, where individual questions are examined in greater depth to gather the experience and position of the experts.

The following questions are based on the three categories corresponding to our research: (1) Lifecycle Data Sharing, (2) Blockchain Suitability and (3) Prototype.

(1) Lifecycle Data Sharing

- What roles and which participants can be found involved in an industrial assets' lifecycle?
- What data is relevant to manage a Digital Twin of an industrial asset?
- Which participants share (respectively require) which type of data?
- Is data currently shared at all about assets?
- What would be potential benefits for the firm that could be realized through sharing data about assets?

(2) Blockchain Suitability

- Is the solution suitable to share data over an industrial asset's lifecycle?
- To what extent do the following aspects benefit the adoption of the solution:
 - Access control,
 - Encryption and
 - Decentralization?

(3) Prototype

- Is the user interface satisfactory? How could the usability be improved?
- What are the current practical requirements regarding throughput and latency:
 - How many Digital Twins are created per month?
 - How many asset-related documents are shared per day?
 - How many sensor data of an industrial asset occur per second?
- Would a self-hosted solution be preferred over a public solution with transaction costs?

References

- Angrish, A., Craver, B., Hasan, M., & Starly, B. (2018). A case study for blockchain in manufacturing: *æfabrecg*: A prototype for peer-to-peer network of manufacturing nodes. *Procedia Manufacturing*, 26, 1180–1192. <https://doi.org/10.1016/j.promfg.2018.07.154>. 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA
- Ayman, A., Aziz, A., Alipour, A., & Laszka, A. (2019). Smart contract development in practice: Trends, issues, and discussions on stack overflow. *CoRR*.
- Badsha, S., Vakilinia, I., & Sengupta, S. (2020). BloCyNfo-Share: Blockchain based Cybersecurity Information Sharing with Fine Grained Access Control. *10th annual computing and communication workshop and conference, {ccwc} 2020, las vegas, nv, usa, january 6–8, 2020* (pp. 317–323). IEEE. <https://doi.org/10.1109/CCWC47524.2020.9031164>.
- Baig, F., & Wang, F. (2019). Blockchain enabled distributed data management - A vision. *35th IEEE international conference on data engineering workshops, ICDE workshops 2019, macao, china, april 8–12, 2019* (pp. 28–30). IEEE. <https://doi.org/10.1109/ICDEW.2019.00-39>.
- Baumgart, I., & Mies, S. (2007). S/Kademlia: A practicable approach towards secure key-based routing. *International conference on parallel and distributed systems* (pp. 1–8). <https://doi.org/10.1109/ICPADS.2007.4447808>.
- Berdik, D., Otoum, S., Schmidt, N., Porter, D., & Jararweh, Y. (2021). A survey on blockchain for information systems management and security. *Information Processing and Management*, 58(1), 102397. <https://doi.org/10.1016/j.ipm.2020.102397>.
- Boschert, S., Heinrich, C., & Rosen, R. (2018). Next Generation Digital Twin. *Proceedings of mme* (pp. 209–217).
- Coyne, E. J., & Weil, T. R. (2013). ABAC And RBAC: Scalable, flexible, and auditable access management. *IT professional*, 15(3), 14–16. <https://doi.org/10.1109/MITP.2013.37>.
- Cruz, J. P., Kaji, Y., & Yanai, N. (2018). RBAC-SC: Role-based access control using smart contract. *IEEE Access*, PP, 1. <https://doi.org/10.1109/ACCESS.2018.2812844>.
- Di Francesco Maesa, D., Mori, P., & Ricci, L. (2019). A blockchain based approach for the definition of auditable access control systems. *Computers & Security*, 84, 93–119. <https://doi.org/10.1016/j.cose.2019.03.016>.

- Dietz, M., & Pernul, G. (2020a). Digital twin: Empowering enterprises towards a system-of-Systems approach. *Business & Information Systems Engineering*, 62(2), 179–184.
- Dietz, M., & Pernul, G. (2020b). Unleashing the digital Twin's potentials for ICS security. *IEEE Security & Privacy*.
- Dietz, M., Putz, B., & Pernul, G. (2019). A distributed ledger approach to digital twin secure data sharing. In S. N. Foley (Ed.), *Data and applications security and privacy xxiii* (pp. 281–300). Springer International Publishing. https://doi.org/10.1007/978-3-030-22479-0_15.
- Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., & Tan, K.-L. (2017). BLOCKBENCH: A Framework for Analyzing Private Blockchains. In *SIGMOD'17 Proceedings of the 2017 ACM international conference on management of data* (pp. 1085–1100). New York, NY, USA: ACM. <https://doi.org/10.1145/3035918.3064033>.
- Eckhart, M., & Ekelhart, A. (2018). Towards Security-Aware Virtual Environments for Digital Twins. *Proceedings of the 4th ACM workshop on cyber-physical system security* (pp. 61–72). <https://doi.org/10.1145/3198458.3198464>.
- Esposito, C., Tamburis, O., Su, X., & Choi, C. (2020). Robust decentralised trust management for the internet of things by using game theory. *Information Processing and Management*. <https://doi.org/10.1016/j.ipm.2020.102308>.
- Ethereum Swarm Contributors (2019). Swarm Documentation. <https://swarm-guide.readthedocs.io/en/latest>.
- Fette, I., & Melnikov, A. (2011). RFC6455 - The WebSocket protocol. *IETF Standards Track*. <https://doi.org/10.17487/RFC6455>.
- Haarmann, S., Batoulis, K., Nikaj, A., & Weske, M. (2018). DMN Decision Execution on the Ethereum Blockchain. In J. Krogstie, & H. A. Reijers (Eds.), *Caise 2018* (pp. 327–341). Cham: Springer International Publishing.
- Hasan, H. R., Salah, K., Jayaraman, R., Omar, M., Yaqoob, I., Pesic, S., ... Boscovic, D. (2020). A blockchain-Based approach for the creation of digital twins. *IEEE Access*.
- Huang, S., Wang, G., Yan, Y., & Fang, X. (2020). Blockchain-based data management for digital twin of product. *Journal of Manufacturing Systems*.
- Kaur, M. J., Mishra, V. P., & Maheshwari, P. (2020). The convergence of digital twin, IoT, and machine learning: Transforming data into action. In M. Farsi, A. Daneshkhah, A. Hosseini-Fa, & H. Jahankhani (Eds.), *Digital twin technologies and smart cities* (pp. 3–17). Springer International Publishing.
- Khan, M. A., & Salah, K. (2018). IoT Security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82, 395–411.
- Kuhn, D. R., Coyne, E. J., & Weil, T. R. (2010). Adding attributes to role-Based access control. *Computer*, 43(6), 79–81. <https://doi.org/10.1109/MC.2010.155>.
- Lazar, J., Feng, J., & Hochheiser, H. (2017). *Research methods in human-Computer interaction, 2nd edition*. Morgan Kaufmann.
- Li, J., Wu, J., Jiang, G., & Srikanthan, T. (2020). Blockchain-based public auditing for big data in cloud storage. *Information Processing and Management*, 57(6), 102382. <https://doi.org/10.1016/j.ipm.2020.102382>.
- López-Pintado, O., Dumas, M., García-Bañuelos, L., & Weber, I. (2019). Dynamic Role Binding in Blockchain-Based Collaborative Business Processes. In P. Giorgini, & B. Weber (Eds.), *Caise 2019* (pp. 399–414). Cham: Springer International Publishing.
- Malakuti, S., & Grüner, S. (2018). Architectural aspects of digital twins in IIoTsystems. *Proceedings of the 12th European Conference on Software Architecture Companion Proceedings - ECSA '18*, 1–2.
- Mandolla, C., Petruzzelli, A. M., Percoco, G., & Urbinati, A. (2019). Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry. *Computers in Industry*, 109, 134–152. <https://doi.org/10.1016/j.compind.2019.04.011>.
- Meroni, G., & Plebani, P. (2018). Combining Artifact-Driven Monitoring with Blockchain: Analysis and Solutions. *Caise 2018* (pp. 103–114). Cham: Springer International Publishing.
- Pedersen, A. B., Risius, M., & Beck, R. (2019). A ten-step decision path to determine when to use blockchain technologies. *MIS Quarterly Executive*, 18(2), 99–115. <https://doi.org/10.17705/2msqe.00010>.
- Ríos, J., Hernández, J. C., Oliva, M., & Mas, F. (2015). Product avatar as digital counterpart of a physical individual product: Literature review and implications in an aircraft. In R. Curran, N. Wognum, M. Borsato, J. Stjepanic, & W. J. C. Verhagen (Eds.), *Advances in Transdisciplinary Engineering 2. Transdisciplinary lifecycle analysis of systems - proceedings of the 22nd ISPE inc. international conference on concurrent engineering, delft, the netherlands, july 20–23, 2015* (pp. 657–666). IOS Press. <https://doi.org/10.3233/978-1-61499-544-9-657>.
- Rouhani, S., Belchior, R., Cruz, R. S., & Deters, R. (2020). Distributed attribute-based access control system using a permissioned blockchain. *CoRR*, *abs/2006.04384*.
- Rubio, J. E., Roman, R., & López, J. (2017). Analysis of cybersecurity threats in industry 4.0: The case of intrusion detection. In G. D'Agostino, & A. Scala (Eds.), *Lecture Notes in Computer Science: 10707. Critical information infrastructures security - 12th international conference, CRITIS 2017, lucca, italy, october 8–13, 2017, revised selected papers* (pp. 119–130). Springer. https://doi.org/10.1007/978-3-319-99843-5_11.
- Saltini, R., & Hyland-Wood, D. (2019). IBFT 2.0: A Safe And live variation of the IBFT blockchain consensus protocol for eventually synchronous networks. *CoRR*, *abs/1909.1*.
- Schroeder, G. N., Steinmetz, C., Pereira, C. E., & Espindola, D. B. (2016). Digital twin data modeling with automationml and a communication methodology for data exchange. *IFAC-PapersOnLine*, 49(30), 12–17.
- Servos, D., & Osborn, S. L. (2016). Strategies for incorporating delegation into attribute-based access control (ABAC). In F. Cappens, L. Wang, N. Cappens-Boulahia, N. Tawbi, & J. García-Alfaro (Eds.), *Lecture Notes in Computer Science: 10128. Foundations and practice of security - 9th international symposium, FPS 2016, Québec city, qc, canada, october 24–25, 2016, revised selected papers* (pp. 320–328). Springer. https://doi.org/10.1007/978-3-319-51966-1_21.
- Tikhomirov, S., Voskresenskaya, E., Ivanitskiy, I., Takhaviev, R., Marchenko, E., & Alexandrov, Y. (2018). SmartCheck: Static Analysis of Ethereum Smart Contracts. *2018 ieee/acm 1st international workshop on emerging trends in software engineering for blockchain (wetebs)* (pp. 9–16).
- Uhlemann, T. H.-J., Lehmann, C., & Steinhilper, R. (2017). The digital twin: Realizing the cyber-physical production system for industry 4.0. *Procedia CIRP*, 61, 335–340. <https://doi.org/10.1016/j.procir.2016.11.152>.The 24th CIRP Conference on Life Cycle Engineering
- Venable, J. R., Pries-Heje, J., & Baskerville, R. L. (2012). A comprehensive framework for evaluation in design science research. In K. Peffers, M. A. Rothenberger, & W. L. K. Jr. (Eds.), *Lecture Notes in Computer Science: 7286. Design science research in information systems. advances in theory and practice - 7th international conference, DESRIST 2012, las vegas, nv, usa, may 14–15, 2012. proceedings* (pp. 423–438). Springer. https://doi.org/10.1007/978-3-642-29863-9_31.
- Xu, X., Pautasso, C., Zhu, L., Lu, Q., & Weber, I. (2018). A Pattern Collection for Blockchain-based Applications. *Proceedings of the 23rd european conference on pattern languages of programs* (pp. 3:1–3:20). ACM.
- Xu, X., Weber, I., & Staples, M. (2019). *Architecture for blockchain applications*. Springer. <https://doi.org/10.1007/978-3-030-03035-3>.
- Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., & Wan, J. (2019). Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*. <https://doi.org/10.1109/JIOT.2018.2847705>.
- Zhao, Q., Chen, S., Liu, Z., Baker, T., & Zhang, Y. (2020). Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems. *Information Processing and Management*. <https://doi.org/10.1016/j.ipm.2020.102355>.