In 2009, blockchain technology was first introduced as the supporting database technology for digital currencies. Since then, more advanced derivations of the technology have been developed under the broader term Distributed Ledgers, with improved scalability and support for general-purpose application logic. As a distributed database, they are able to support interorganizational information sharing while assuring desirable information security attributes like non-repudiation, auditability and transparency. Based on these characteristics, researchers and practitioners alike have begun to identify a plethora of disruptive use cases for Distributed Ledgers in existing application domains. While these use cases are promising significant efficiency improvements and cost reductions, practical adoption has been slow in the past years. This dissertation focuses on improving three aspects contributing to slow adoption. First, it attempts to identify application areas and substantiated use cases where Distributed Ledgers can considerably advance the security of information sharing. Second, it considers the security aspects of the technology itself, identifying threats to practical applications and detection approaches for these threats. And third, it investigates success factors for successful interorganizational collaborations using Distributed Ledgers.

Universität Regensburg
UNIVERSITÄTSBIBLIOTHEK

ISBN 978-3-88246-473-3
9 783882 464733

Secure Information Sharing with Distributed Ledgers

Benedikt Putz

Benedikt Putz

Secure Information Sharing with Distributed Ledgers

Eine Publikation der
Universitätsbibliothek Regensburg

Universität Regensburg
Fakultät für Wirtschaftswissenschaften
Lehrstuhl für Wirtschaftsinformatik I - Informationssysteme

Secure Information Sharing with Distributed Ledgers

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaft

eingereicht an der Fakultät für Wirtschaftswissenschaften
der Universität Regensburg

vorgelegt von:
Benedikt Putz, M.Sc. with Honors

Berichterstatter:
Prof. Dr. Günther Pernul
Univ.-Prof. Dipl.-Ing. Mag. Dr. techn. Edgar Weippl

Tag der Disputation: 29.06.2022

Erstgutachter (Betreuer): Prof. Dr. Günther Pernul

Zweitgutachter: Prof. Dr. Edgar Weippl

Tag der Prüfung: 29 Juni 2022

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie. Detaillierte bibliografische Daten sind im Internet unter http://dnb.dnb.de abrufbar.

ISBN (Print): 978-3-88246-473-3

ISBN (PDF): 978-3-88246-474-0

DOI: 10.5283/epub.52636

Link zur Online-Version: https://epub.uni-regensburg.de/52636

Satz und Layout: Benedikt Putz

Umschlaggestaltung: Peter Brünsteiner

Herstellung: Universitätsbibliothek Regensburg

Erscheinungsort: Regensburg, 2022

Druck und Bindung: Digital Print Group o. Schimek GmbH, Nürnberg

Acknowledgement

For the past three years, many people have supported me on my way to finishing this cumulative dissertation. While it may appear as a single book in print, it is in fact the result of countless hours of discussions, idea exchanges and co-programming/writing with fellow researchers and friends. This page is dedicated to these people.

First, I'd like to thank my supervisor Prof. Dr. Günther Pernul for the continued support and feedback on my research. Several multi-faceted discussions about the topic of blockchain eventually lead to my sincere belief that blockchain and DLT can and will improve this world, despite their public misperception as mere subjects of financial speculation. Besides the professional guidance towards finding a research topic and focus, his mentoring was also invaluable to my personal development. In addition, I would like to thank my secondary supervisor Prof. Dr. Weippl for the advice and feedback on the research topic and method.

Second, but no less important, I'd like to thank the IFS research team for all the support, collaboration and recreational fun. Flori, for the mentoring during my Master's thesis, subsequent publications and the countless discussions about crypto and blockchain (speculation, crypto dogs, or otherwise). Petra and Werner, for all the support and coffee talks. Fabi, for the honest interest in my work, our startup ideas and of course for all the memes. Fred, for being there during COVID times and for the extensive survival bike tours. Marietheres, for the extensive and prolific research collaboration and for introducing me to Yoga. Dani, for perservering throughout the years as my office mate and workout partner. Philip, for the contribution to EtherTwin and the thoroughly entertaining Greece trip. Ludwig and Sabrina, for supporting me in the TRIO project, but also on Strava. Lena, for the motivation, the feedback on this dissertation, and the many enjoyable hours out climbing.

Third, I am thankful for the professional support and social community of the Honors program at the University of Regensburg, initiated by Prof. Dr. Michael Dowling. Without the Honors program, I would not have started the PhD project in the first place. Throughout the years I took on various roles as student, webmaster, coordinator and alumnus.

Finally, this work is dedicated to my parents Josef and Angelika, for being the foundation I can count on to climb ever taller mountains.

Abstract

In 2009, blockchain technology was first introduced as the supporting database technology for digital currencies. Since then, more advanced derivations of the technology have been developed under the broader term Distributed Ledgers, with improved scalability and support for general-purpose application logic. As a distributed database, they are able to support interorganizational information sharing while assuring desirable information security attributes like non-repudiation, auditability and transparency. Based on these characteristics, researchers and practitioners alike have begun to identify a plethora of disruptive use cases for Distributed Ledgers in existing application domains. While these use cases are promising significant efficiency improvements and cost reductions, practical adoption has been slow in the past years. This dissertation focuses on improving three aspects contributing to slow adoption. First, it attempts to identify application areas and substantiated use cases where Distributed Ledgers can considerably advance the security of information sharing. Second, it considers the security aspects of the technology itself, identifying threats to practical applications and detection approaches for these threats. And third, it investigates success factors for successful interorganizational collaborations using Distributed Ledgers.

Contents

List of Figures

List of Tables

Part I

Outline of the dissertation

1 Motivation

In a highly interconnected world, information sharing among collaborating organizations has become a necessity. Complex supply chains require several months to years of advance planning, making timely information sharing critical toward meeting the stakeholders' needs. Information asymmetry resulting from poor sharing practices can lead to shortages, inefficiency and overall poor supply chain performance [18, 57]. This is apparent in application domains like agrifood [17, 58], where transparency, provenance, and traceability are important to consumers. Similar information sharing requirements exist in other sectors, including industrial supply chains, financial services, government services, and healthcare [5].

Existing information systems have limitations in addressing the requirements of these information sharing practices, especially concerning security aspects. Current practices rely on two approaches: ad-hoc exchange and trusted third parties. For ad-hoc exchange, standards such as EDI (electronic data interchange) are commonly used. There are several issues with such standards: connections are bilateral only (unicast), there is no single valid interaction history, and it is difficult to assure integrity [21]. Using trusted third parties (i.e. cloud providers) solves these problems, but requires complete trust in the third-party provider regarding both organizational and technical elements. Confidentiality, integrity and availability of the information are at the hands of the third-party provider. In times where data breaches, malware infections and denial of service attacks are widespread [16], such trust can quickly prove unfounded if the aforementioned security goals are compromised while the data is stored at the third party. In fact, information security threats, unauthorized access and cyberattacks are common concerns in supply chain information sharing [35].

Distributed Ledger Technology (DLT) is a recently introduced technological artifact that promises to solve these information sharing issues. First introduced with the proposal of the Bitcoin blockchain in 2008 [45], *blockchain* technology has since evolved to cover additional use cases beyond cryptocurrencies. Since the term *blockchain* no longer fits all instantiations of the technology, the more abstract term *Distributed Ledger* has been introduced.

Distributed Ledgers are replicated databases, kept up to date by a network of peers using a consensus algorithm. The key novelty compared to traditional databases is that

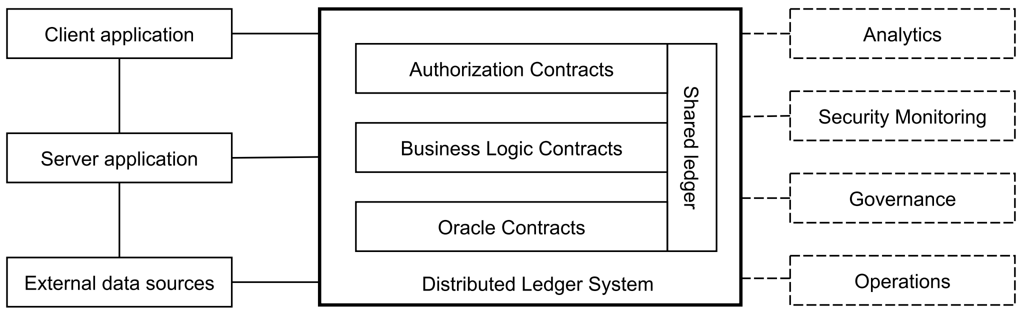


Figure 1: Architecture and integration of an enterprise DLT application.

transaction updates are stored as an append-only sequence of blocks. These blocks are immutably linked through a hash-chain, where each block includes the predecessor's block hash. Each peer permanently stores all transactions starting with the first block. If the consensus algorithm tolerates byzantine faults, the network can be operated by parties that do not fully trust one another, given an honest majority [55]. Transactions are observed and agreed on by all participants, which means updates are non-repudiable. Business logic is executed verifiably through *smart contracts*. Additional supportive contracts provide authorization and external data integration (referred to as *oracles*). Besides the Distributed Ledger itself, DLT applications, like traditional web applications, use client and server applications to integrate with existing systems. In addition, DLT applications are supported by organizational functions, which ensure stable and secure operations, governance and analytical insights. An overview of a typical enterprise DLT application is given in Figure 1.

Despite their promise, Distributed Ledgers have not yet reached widespread adoption for secure information sharing. At the time of this writing, there are still few examples where Distributed Ledgers are successfully used in business operations (i.e. Tradelens [30], we.trade [28]). Both technical and organizational factors contribute to the slow adoption and development of DLT. On the one hand, several unsolved or only partly solved technical problems regarding scalability, security and privacy remain to date [40]. On the other hand, unlike most traditional enterprise software, DLT applications must be developed in collaboration with other organizations in the same industry to realize their full potential. This leads to organizational challenges regarding collaboration, decision-making and responsibilities [36]. *Information Systems Research* (ISR) is uniquely suited to solve such challenges arising at the intersection of organization and technology.

Hence, using ISR methodology, this dissertation aims to develop applications and concomitant foundations for secure interorganizational information sharing based on DLT. To highlight the relevance and importance of the research conducted in this dissertation and to point out influencing works, Section 2 gives an overview of related research concerning DLT and its applications. Section 3 establishes the structure of the research work by outlining the main research questions, the three derived research areas and the corresponding business problems addressed in the research. Section 4 details the papers published as part of the cumulative dissertation, and how they address the research questions and business problems.

2 Related research

Since the inception of Bitcoin in 2008, numerous DLT frameworks have been developed. While at first many alternatives were derivative cryptocurrencies with only minor modifications, later frameworks focused on developing additional features and improving performance. The first milestone on this path was the development of Ethereum in 2014, the first blockchain supporting complex application logic through Turing-complete *smart contracts* [64]. Smart contracts enable the verifiable execution of business logic on decentralized ledgers without intermediaries. This innovation allows interorganizational information exchange without ceding computation or data sovereignty to a single trusted partner or intermediary. To make smart contracts accessible to users, *decentralized applications* emerged as the decentralized counterpart to traditional web applications [68, 63]. Other novel DLT frameworks like IOTA [49] focused on alternative data structures, using directed acyclical graphs as opposed to chains of blocks to improve scalability. Another approach to improve scalability is consensus protocol innovation, where research developed novel protocols like Proof of Stake [33]. The aforementioned frameworks are referred to as *permissionless*, meaning that they do not restrict participation to a fixed set of nodes. Permissionless blockchains such as Bitcoin and Ethereum inspired the notion of *permissioned* blockchain networks, where a limited set of authorized participants operates the network [66]. Limiting the set of participants is ideal for enterprise applications that do not want to share data beyond the network of business partners. Thus, starting in 2016, several open-source frameworks were developed for permissioned blockchains. The most widely used frameworks in enterprises [51, 61] are Hyperledger Fabric [3], Corda [22], the Bitcoin-based Multichain [19] and the Ethereum-based Quorum [31]. In research, permissionless Ethereum clients are commonly used for prototypes [34], with Quorum offering additional privacy-preserving features such as private transactions and smart contracts.

Based on these DLT frameworks, numerous applications have been developed by researchers [7, 26]. Financial use cases such as cryptocurrencies and decentralized finance were the first ones to be investigated by research and practice [42]. Supply chain and logistics traceability is another major area where DLT is aptly suited to advance digitization [30]. Other research areas of interest include health [13], education [2] and the Internet of Things (IoT) [52]. Within the scope of this dissertation, two application areas are of particular interest for the development of novel applications: the industrial IoT and Cybersecurity. As a subset of IoT research, the industrial IoT is particularly concerned with the security and analytics of industrial data. Thereby, the Digital Twin is a novel concept to map physical objects to digital counterparts [14]. At the start of this dissertation, DLT-based twins had not yet been investigated as a possible solution to the aforementioned challenges [56]. Therefore, this dissertation studies the suitability of DLT as a solution for sharing Digital Twin data in the industrial IoT. In Cybersecurity research, sharing information about threats is essential to improve preparedness for future attacks [41]. However, during the first years of DLT application research, the sharing of Cybersecurity-related information using DLT was studied by comparatively few researchers [54]. Therefore, DLT-based Cybersecurity information sharing presents another research opportunity and focus area for this dissertation.

To build DLT applications for security purposes, the security of DLT itself is paramount. DLT security research extends to all layers of a Distributed Ledger. The most common four-layer model includes the *Network*, *Consensus*, *Replicated State Machine* and *Application* layers [25]. Each of these layers is subject to threats from malicious actors. Attacks differ based on the DLT framework used, but most research has focused on popular blockchain frameworks like Ethereum [27, 50]. Most attacks unique to the blockchain ecosystem focus on the *Replicated State Machine* layer, specifically smart contract programming languages and the corresponding virtual machines. Ethereum's Solidity and the associated Ethereum Virtual Machine are the most popular and widely used smart contract programming environment [4]. Due to the intricacies of smart contract development, they have been widely exploited in attacks on permissionless blockchains [8] and still today represent an attractive target for attackers [32]. Hence, a plethora of primary studies and surveys have been conducted concerning smart contract vulnerabilities in recent years [50]. Besides documenting vulnerabilities, researchers have focused on programming best practices, automated vulnerability detection tools, and both proactive and reactive mitigation efforts [8]. Consequently, mitigation tools for these attacks are subject to research [59, 46, 9]. Other attack examples beyond smart contracts include attacks related to social engineering [29] and attacks on network communication such as denial of service [38] and eclipse attacks [23]. Besides Ethereum, Hyperledger Fabric security has been focused by researchers, as it is the most popular permissioned blockchain platform [11]. Its smart contracts, also called chaincodes, have been the subject of several studies [12, 69]. Due to the use of established programming languages (Go, Node.js) and the lack of cryptocurrency as an incentive for attacks, fewer vulnerabilities are known and exploited. Nevertheless, a significant amount of both internal and external attack vectors exist [11]. Despite these threats, research regarding their detection and mitigation is still sparse. Therefore, this dissertation focuses on understanding threats to permissioned networks, while developing approaches for detection and mitigation. Due to its popularity, Hyperledger Fabric serves as the primary evaluation target.

To build secure applications based on DLT, solid technological and organizational foundations are needed to disincentivize and prevent malicious attacks. There is a large body of literature regarding the architecture of blockchain-based applications [66, 68, 63] and corresponding software patterns [67, 62]. However, there is still a need for research at the intersection of organization and technology [53]. At this intersection, blockchain governance research is concerned with the distribution of responsibilities and power among blockchain network participants [73]. For enterprise blockchain applications, these participants are organizations, which organize as part of a *consortium* [71]. While research exists on the formation of business consortia [37], little is still known about the formation and development of consortia focused on information sharing with DLT beyond isolated case studies [30, 71]. To this end, this dissertation also considers factors for stable and secure interorganizational relations to underpin networked DLT applications.

3 Research Questions

As an emerging technology, DLT opens up a plethora of new research opportunities. Section 2 has given an overview of these new research streams, while highlighting challenges that remain on the path to adoption of secure DLT applications. To address remaining research gaps, this dissertation focuses on developing applications for secure interorganizational information sharing based on secure DLT foundations. This objective leads to the main research question:

Main Research Question. How can organizations apply Distributed Ledger Technology to improve the security of interorganizational information sharing (IIS)?

This research question points toward several perspectives on DLT adoption. The term *"apply"* implies a focus on novel DLT-based applications. These may supplement or replace existing information sharing systems to reach benefits like non-repudiation, improved process efficiency, or improved transparency. *"improving the security"* of IIS implies designing secure systems and protecting these systems from threats. DLT's built-in cryptographic mechanisms enable enhanced security, but the realization of these benefits depends on the proper security foundations and processes of the adopting organizations. The subject matter of *"interorganizational information sharing"* points to the need to securely collaborate with other organizations to gain the desired application benefits.

Based on these considerations, the main research question is segmented into three focus research questions **RQ1**, **RQ2** and **RQ3**. These focus research questions are addressed in publications as part of the cumulative dissertation.

RQ1. How can DLT improve the security of IIS in existing application domains?

RQ2. How can DLT applications be protected from internal and external threats?

RQ3. How can organizations securely share information as part of a DLT consortium?

The research questions logically build upon one another. RQ1 represents the application of a novel technological artifact (DLT) towards existing research areas and business problems. This represents the *Improvement* strategy of the Design Science Research Knowledge Contribution framework [20], i.e. developing knowledge by developing new solutions for existing problems. RQ2 ensures the technical security of the applications developed for RQ1 by modeling applicable threats. RQ3 looks at the longer-term strategic development perspective of the consortium required to support functional and secure DLT applications.

3.1 Methodology

The research conducted in this dissertation is part of the domain of Information Systems Research (ISR). Publications in ISR mostly follow one of two complementary research paradigms: behavioral science and design science. Behavioral science develops theories to explain and predict human and organizational behavior. Design science seeks to create novel Information Systems (IS) artifacts to solve practical problems. [24]

To address the challenges posed by the main research question, design science is aptly suited. Solving technical challenges and demonstrating improvement over existing systems necessitates the development of IS artifacts. Thus, the main research method utilized for the design of the technological artifacts is the Design Science Research (DSR) paradigm for ISR [24]. More specifically, the DSR methodology (DSRM) process model [48] is utilized in many works of this dissertation and shown in Figure 2.

To address the organizational aspects of secure foundations for DLT information sharing (RQ2), Case Study Research methods present an adequate tool [70].

3.2 Research Process

To answer the research questions, the dissertation follows the rigorous 6-step iterative process model of the DSRM [48] shown in Figure 2. Its steps and their relevance to problem-solving in the DLT context are outlined below.

Problem identification. Based on current trends in industry and academia, distinct contemporary business problems with respect to the research questions are identified.

Objective definition. The artifact objective is defined based on the benefits DLT can bring to the application context over existing solutions (RQ1). For RQ2, the artifact objective is centered around improving DLT security, while for RQ3 it focuses on improving interorganizational collaboration towards the successful application of DLT.

Design & development. The research artifact is designed based on the requirements demanded by the objective. Development of an instantiated artifact is performed if an implementation is needed for the following stages. The implementation follows secure-by-design principles [39].

Demonstration. The artifact is applied to the domain problem to demonstrate its ability to solve the problem and to do so more effectively and/or efficiently than previous solutions.

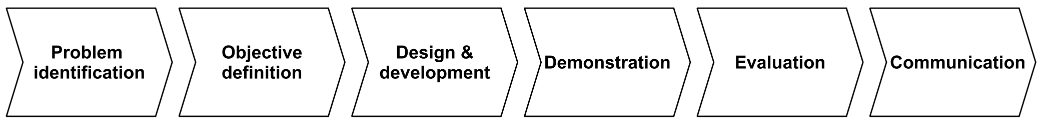


Figure 2: DSRM according to Peffers et al. [48]

Evaluation. The evaluation determines how well it meets the objectives established in step 2. For qualitative measurement, semi-structured expert interviews are used to assess how well the artifact addresses the problems of the intended user group. For quantitative measurement, performance-related technical characteristics of the prototype are benchmarked.

Communication. The act of publishing research results in scholarly publications is an explicit objective of this cumulative dissertation. Appropriate venues are chosen based on the context of the addressed DLT problem. To ensure full research transparency, the artifacts are published open-source if an implementation was performed.

3.3 Research plan

To address RQ1 through RQ3, specific business problems (BP) are identified within the problem space of each research question, corresponding to the *Problem Identification* step of the DSRM outlined in the previous section. These problems represent contemporary challenges that are currently faced by researchers or industry, which are in need of a new or improved solution. This dissertation performs research actions to respond to the challenges, yielding a single or several publications per business problem as part of the *Communication* step.

Main RQ. How can organizations apply Distributed Ledger Technology to improve the security of interorganizational information sharing (IIS)?

RQ1. How can DLT applications improve the security of IIS in existing application domains?

- **BP1-1** How can Distributed Ledgers improve the security of industrial IIS?
- **BP1-2** How can Distributed Ledgers improve IIS for cybersecurity?

RQ2. How can DLT applications be protected from internal and external threats?

- **BP2-1** What internal and external threats are Distributed Ledgers susceptible to?
- **BP2-2** How can attacks on Distributed Ledgers be detected?

RQ3. How can organizations securely share information as part of a DLT consortium?

- **BP3-1** Is there a lifecycle for the development of DLT applications for IIS in a consortium?

---

4 Results

The research questions proposed in Section 3 were addressed in a total of eight publications. A complete publication overview is given in Table 1, including the ranking of the respective conference or journal in the CORE rankings[1].

Table 1: Publication overview.

No	Title	CORE
P1	Marietheres Dietz, Benedikt Putz, and Günther Pernul. 2019. A Distributed Ledger Approach to Digital Twin Secure Data Sharing. In Data and Applications Security and Privacy XXXIII, Simon N. Foley (ed.). Springer International Publishing, 281–300.	B
P2	Benedikt Putz, Marietheres Dietz, Philip Empl, and Günther Pernul. 2021. Ethertwin: Blockchain-based secure digital twin information management. Information Processing & Management 58, 1 (2021).	A
P3	Florian Menges, Benedikt Putz, and Günther Pernul. 2020. DEALER: decentralized incentives for threat intelligence reporting and exchange. International Journal of Information Security (2020).	C
P4	Benedikt Putz, Florian Menges, and Günther Pernul. 2019. A secure and auditable logging infrastructure based on a permissioned blockchain. Computers & Security 87, (November 2019), 101602.	B
P5	Benedikt Putz and Günther Pernul. 2019. Trust Factors and Insider Threats in Permissioned Distributed Ledgers. Transactions on Large-Scale Data- and Knowledge-Centered Systems XLII, (2019), 25–50.	-
P6	Benedikt Putz and Günther Pernul. 2020. Detecting Blockchain Security Threats. In 2020 IEEE International Conference on Blockchain (Blockchain), IEEE, 313–320.	-
P7	Benedikt Putz, Fabian Böhm, and Günther Pernul. 2021. HyperSec: Visual analytics for blockchain security monitoring. In ICT systems security and privacy protection - 36th IFIP TC 11 International Conference, SEC 2021, Oslo, Norway, June 22-24, 2021, Proceedings (IFIP Advances in Information and Communication Technology), Springer, 165–180.	B
P8	Benedikt Putz and Günther Pernul. 2022. Comparing Successful DLT Consortia: A Lifecycle Perspective. In 55th Hawaii International Conference on System Sciences 2022, 4591–4600.	A

Each publication addresses a specific business problem from Section 3.3. For some business problems multiple publications address distinct sub-problems. It should be noted that these publications do not fully exhaust the problem space of the research question. As is common in research, other complementary publications exist that address similar sub-problems. In addition to the Related Work chapter within each publication, a summary

[1]http://portal.core.edu.au

of other related research is given in Section 2. Nevertheless, each publication in Table 1 provides a significant contribution towards solving the larger business problem and its superordinate research question.

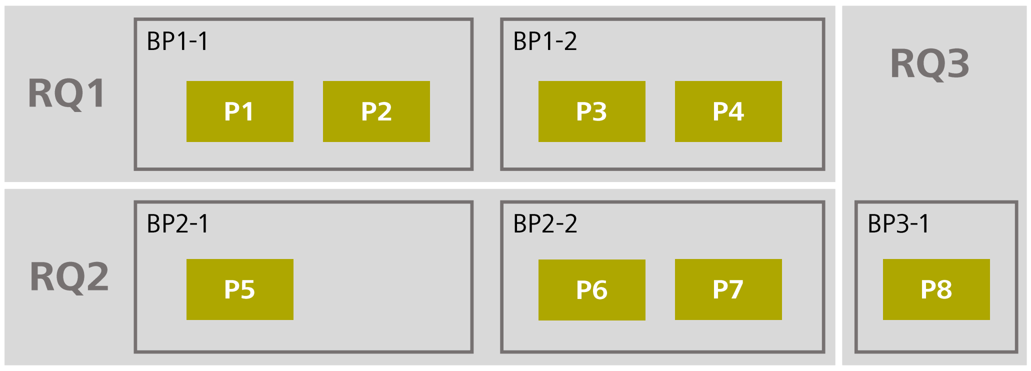


Figure 3: Mapping of research questions to business problems (BP*) and publications (P*).

4.1 RQ1: Novel DLT applications for secure information sharing

Distributed Ledgers go beyond existing distributed databases by providing a shared storage platform without intermediaries. Availability, integrity and non-repudiation guarantees are built into the technology through a verifiable replicated data structure. These properties make it an attractive candidate to replace existing information sharing applications based on bilateral exchange or intermediaries. For reasons mentioned in Section 2, this dissertation focuses on applying DLT-based information sharing to two application areas: the **Industrial Internet of Things (IIoT)** and **Cybersecurity**.

The IIoT is an emerging network of internet-connected devices and machines, which interact across industrial organizations to share data and create business value. It is in need of a secure way to store and share data across organizational boundaries, without intermediaries and while maintaining privacy as needed. Publications P1 and P2 focus on the development of DLT-based artifacts for this purpose (cf. **BP1-1**).

As the digitization of society advances, the number and impact of cybersecurity threats have kept pace. Correspondingly, researchers have focused on developing more efficient means to anticipate, prevent and mitigate threats. As a secure information sharing technology, DLT can help store and share cybersecurity-related data while providing availability and integrity guarantees. P3 and P4 focus on applying DLT to improve IIS for cybersecurity (**BP1-2**). With regard to the anticipation of threats, Publication P3 demonstrates how it can be applied to sharing Cyber Threat Intelligence (CTI) among organizations. To support mitigation and prosecution of cybersecurity threat actors, Publication P4 develops a DLT solution for non-repudiation of log entries. Immutable time-stamped log entries can then be used as evidence for the prosecution of attackers in court.

4.1.1 A distributed ledger approach to digital twin secure data sharing [P1]

P1 develops a framework to securely share Digital Twin data using DLT. The Digital Twin paradigm refers to a digital representation of a real-world asset. Throughout this lifecycle, the twin's data needs to be shared with various stakeholders – e.g. manufacturer, owner, distributor and maintainer. With conventional sharing techniques it is challenging to share this data securely, so a DLT-based approach is proposed to fulfill all requirements.

The article follows the DSRM by first formally defining the problem as part of the *Problem Identification* step. The formal definition is based on ensuring two main goals: data confidentiality (authentication, authorization) and integrity (auditability, traceability). For *Objective Definition*, five requirements are derived from the formal basis and practical challenges:

R1 Multi-party sharing

R2 Data variety support

R3 Data velocity support

R4 Data integrity and confidentiality mechanisms

R5 Read and write operations

R2 and R3 originate in the different data formats and throughput requirements needed for IoT data, which can be challenging to handle for data sharing systems. Thereafter, in the *Design & Development* stage, appropriate technologies are selected, and a system architecture is developed. A *Demonstration* is not part of P1 and instead performed in the follow-up publication P2, which builds on the results of P1. A critical evaluation and discussion of the resulting architecture provide directions for future research.

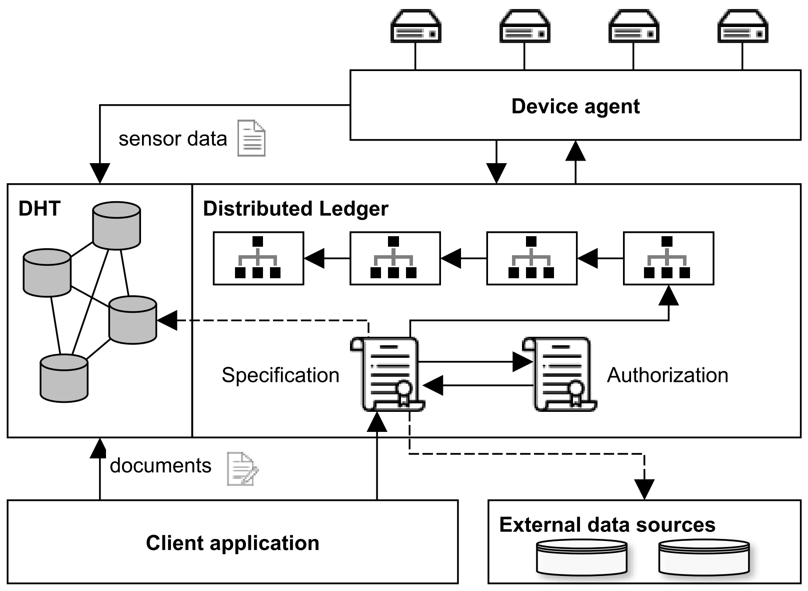


Figure 4: Architecture framework for digital twin data sharing using distributed ledgers.

The proposed application architecture is shown in Figure 4. After evaluating if DLT is a good fit for the use case using the framework by Wüst and Gervais [65], a Distributed Ledger is chosen as a decentralized storage platform. It stores the specification metadata about the physical asset, as well as references to all other shared documents on external systems. Access control is enabled through a dedicated Authorization smart contract. To store and share larger documents, a Distributed Hash Table (DHT) is introduced as off-chain storage. It also enables low latency sharing of sensor data, which avoids the longer latencies associated with DLT transactions. Users interact with the Digital Twin using a client application, where they can modify specification data and upload documents. A device agent connects the Digital Twin to its physical counterpart. Sensor data is provided by this device agent, which monitors the physical asset and pushes measurements to the DHT. The concept also allows for two-way interaction, for example through PLC program calls issued on the Digital Twin, which are then forwarded to the asset by the device agent using a pull model.

The intended functionality of the architecture is demonstrated as part of a bottling plant use case. The use case highlights the advantages of the decentralized solution, such as improved transparency and integrity guarantees. These are enabled by the lack of a trusted third party, which might manipulate the data for personal gain.

Contribution of P1: The paper introduces a novel framework for secure Digital Twin data sharing. Based on a definition of practical requirements for multi-party sharing, the applicability of DLT to Digital Twin data sharing is evaluated and affirmed. A distributed architecture is developed using DLT-based smart contracts and DHT off-chain storage. The suitability of the design is evaluated in an exemplary industrial use case, demonstrating the fulfillment of the requirements.

4.1.2 Ethertwin: Blockchain-based secure digital twin information management [P2]

P2 builds on the results of P1 by implementing a prototype for Digital Twin information management and sharing on a blockchain. The goal is to demonstrate the practical feasibility by implementing a fully working prototype. The publication follows the DSRM, utilizing the results of P1 with regard to Problem Identification and Objective Definition. A ten-step decision path [47] is used to evaluate the applicability of blockchain to the DT use case. For application design, P2 uses the requirements established in P1 to develop a component diagram and detailed entity-relationship data model (ERM) (see Figure 5) for the industrial data sharing use case. To ensure data integrity and confidentiality, a fine-grained access control model is developed and implemented as part of a smart contract system. The resulting prototype is published open-source on GitHub[2] and made available publicly for demonstration purposes[3]. It is evaluated in quantitative technical experiments and qualitative expert interviews with industry professionals.

The ERM is shown in Figure 5. The on-chain component implements the access control model using the Specification and Authorization smart contracts shown in Figure 4. A

[2]https://github.com/sigma67/ethertwin
[3]https://ethertwin.ur.de

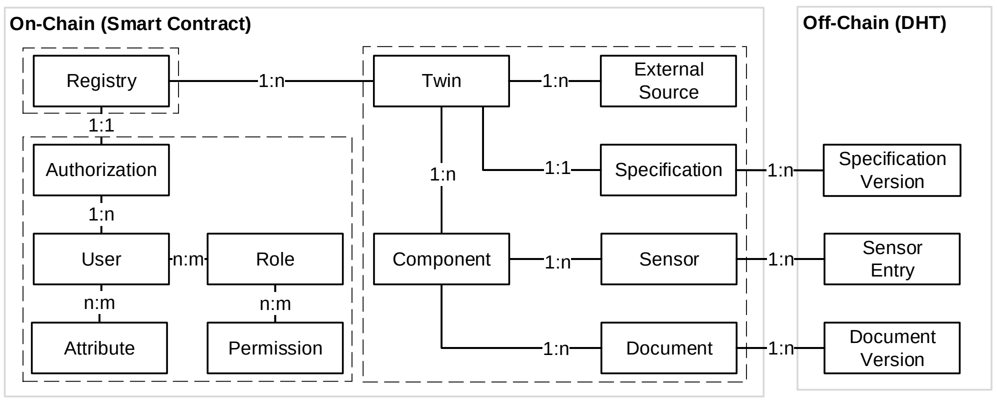


Figure 5: Entity relationship data model of EtherTwin.

Contract Registry keeps track of all deployed Specification contracts (1:n relation), which allows creating multiple *Twins* with a single deployment. Each *Twin* is mapped to a single *Specification*, which provides information about its *Components*. Each *Component* can be associated with *Sensor* data and *Documents*. The Authorization contract implements the access control model and maps *Users* to *Attributes* and *Roles*. The off-chain component is used to store the full data associated with the versioned references stored in the Specification smart contract.

The fine-grained access control model is based on the RBAC-A model, which describes a role-centric model enhanced with attributes [10]. The base permissions are defined by roles, which can be limited by the user's attributes. Write permissions are enforced on-chain by limiting which transactions a user can perform when interacting with the Specification contract. Read permissions are enforced off-chain, by encrypting data stored on the DHT with a file key. This key is provided to all participants using their blockchain public key identities based on a key distribution algorithm specified in the paper.

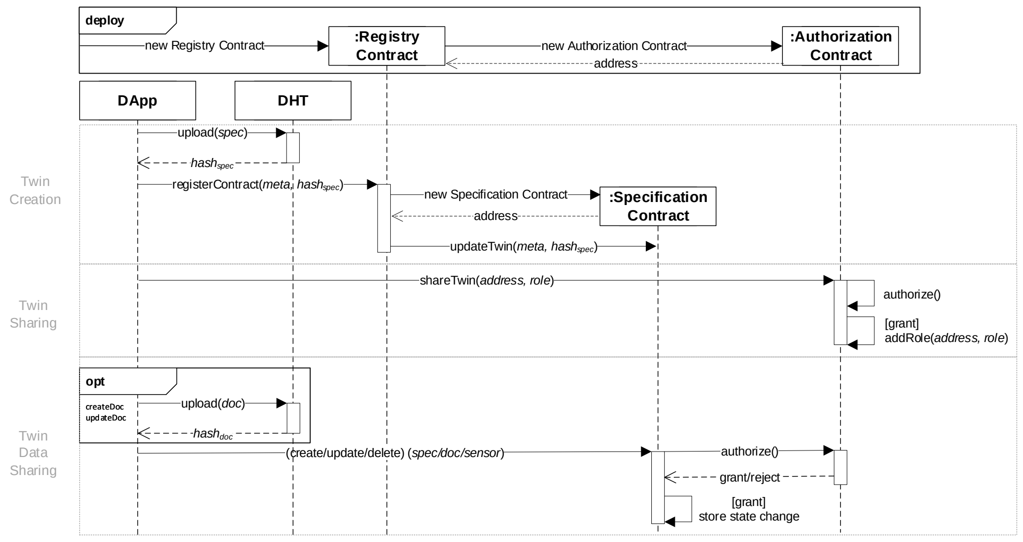


Figure 6: Sequence diagram of write interactions in EtherTwin.

Two complementary technologies were chosen for the open-source implementation: the Ethereum blockchain framework and the Swarm DHT. The Ethereum blockchain is widely used in research due to its popularity and developer tools, enabling swift creation and evaluation of prototypes. We use it to implement our smart contract system. The Swarm DHT is well integrated with Ethereum, as it uses the same public key identities based on the Elliptic Curve Digital Signature Algorithm (ECDSA). This facilitates the integration of both systems and the implementation of the access control model. A sophisticated frontend was developed using the single page application framework Vue.JS[4]. Figure 6 shows the complex interactions among components for the write operations Twin Creation, Twin Sharing and Twin Data Sharing. In total there are five distinct components: three smart contracts, the DApp (frontend) and the DHT.

Technical evaluation of the prototype is performed using quantitative measurements of transaction cost and algorithmic latency. The results show that an implementation in an industry use case is feasible, although transaction costs may be prohibitively high when using the public Ethereum main net due to cryptocurrency exchange rates. A private blockchain can provide a cost-efficient alternative. Qualitative evaluation is performed as part of a use case with an industry partner, as well as structured expert interviews with domain experts from the industry. The prototype's user interface and the use of access control and encryption was lauded by the experts. There was some skepticism regarding the scalability of a decentralized system when used with the large volumes of sensor data occurring in practice. Overall, the experts considered the prototype a good starting point for enabling industrial information sharing based on Digital Twins.

Contribution of P2: P2 develops EtherTwin, an open-source decentralized application prototype designed to address the requirements of multi-party industrial data sharing using Digital Twins. It follows an owner-centric sharing model involving all lifecycle participants, while ensuring appropriate access with a fine-grained RBAC-A access control model. The EtherTwin prototype shows how to overcome practical challenges associated with decentralized data sharing, while addressing throughput and latency requirements at manageable cost.

4.1.3 DEALER: Decentralized Incentives for Threat Intelligence Reporting and Exchange [P3]

With the advance of digitization, cybersecurity threats have been proliferating along with the increasing use of information systems. Sharing information about existing threats improves organizations' awareness of the current threat landscape, and can be used to better defend against future attacks (voluntary reporting). However, the sensible nature of threat information often has an inhibitory effect on an organization's willingness to share its incidents. A sustainable platform for CTI exchange should provide such adequate incentives. Besides voluntary reporting, obligatory reporting is mandated for critical infrastructure providers by regulatory requirements such as the IT-Sicherheitsgesetz[5]. To fulfill these

[4]https://vuejs.org
[5]https://www.gesetze-im-internet.de/bsig_2009/BJNR282110009.html

Table 2: Comparison of CTI sharing platforms with regard to reporting and sharing requirements. ○ not addressed, ◐ insufficiently addressed, ● explicitly addressed

	FB-TX	X-Force	MISP	OpenCTI	Trident	DEALER
Platform availability	◐	●	◐	◐	●	●
Data availability	●	●	●	●	○	◐
Integrity	○	○	○	○	◐	●
Non-repudiation	○	○	○	◐	●	●
Incentives	○	○	○	○	●	●
Fairness	○	○	○	○	◐	●
Quality Assurance	○	○	○	○	◐	◐

regulatory requirements, a CTI exchange platform must provide availability, integrity and non-repudiation for stored data. Trusting a third-party provider to operate such a platform is not ideal, as they may violate data confidentiality or seek to profit from the stored data through reselling. To avoid such a trusted third party, P3 proposes the DEALER threat intelligence sharing platform based on DLT. DEALER is designed to provide decentralized incentives for sharing data, while preserving the confidentiality of shared threat intelligence through encryption.

The paper follows the DSRM by first defining problem and objective, as detailed in the previous paragraph. Thereafter, P3 defines seven requirements based on the need for incentives and regulation-compliant reporting. These requirements are shown in Table 2 and subsequently addressed in the design of the DEALER platform. To demonstrate the design, an appropriate blockchain platform and distributed storage technology are selected to implement a decentralized application prototype. For evaluation, the demonstrator is then shown to information security experts in semi-structured expert interviews.

The design of a platform for incentivized data exchange involves **a)** developing mechanisms to protect the data from unauthorized access, and **b)** ensuring fairness for both trading parties. Smart contracts can be used to implement such protocols without a trusted third party [15]. DEALER uses smart contracts for the escrowed sale of data. To protect data confidentiality, all data is encrypted and only made accessible to authorized participants (i.e. sellers and buyers) using off-chain file keys. Fairness is ensured through dispute resolution protocols and sign-up collateral, which may be claimed in case of misbehavior. Besides buyers and sellers, verifiers ensure the data quality of newly uploaded threat intelligence. They also act as dispute resolution agents in case of disagreement over a purchase.

The design is implemented using the permissionless blockchain platform EOS[6]. EOS is chosen mainly due to its unique property of not requiring transaction costs. Instead, it only requires staking currency in return for a daily transaction allotment. Paying a fee for each transaction could disincentivize sharing threat intelligence and threaten cost fairness for the participants. For data storage, the DHT network IPFS is used to share the encrypted incident data and encryption keys. The evaluation of the prototype shows reasonable staking costs for execution of the smart contract on the EOS blockchain, even when scaled to all institutions

[6]https://eos.io

with legal incident reporting obligation. The measured latencies for write interactions are on the order of one to two seconds, which is high but still tolerable. Semi-structured interviews with security experts yield positive results regarding the platform's incentive structure, integrity features and usability. A detailed security discussion describes how various threats to the platform are dealt with (free-riding verifiers, content reselling, verifier collusion and sybil attacks).

Contribution of P3: DEALER provides a decentralized marketplace for the incentivized sharing of threat intelligence, fulfilling both voluntary and obligatory sharing needs. Quality assurance and dispute resolution is provided by independent verifiers. The platform provides transactional fairness for both seller and buyer. An open-source prototype demonstrates the concept's feasibility in a fully decentralized setting using the EOS blockchain and IPFS decentralized storage.

4.1.4 A secure and auditable logging infrastructure based on a permissioned blockchain [P4]

Log data is another form of security information that is ubiquitous today. It is produced by most enterprise applications and often provides critical information for identifying the source of a security incident. While sharing the contents of the logs is undesirable due to sensitive information within, sharing a proof of existence can prove useful for non-repudiation. An immutable time-stamped proof can later be used to prove that the contents of the log entry have not been modified, which can provide indisputable proof of the origin and procedure of an attack (i.e. IP address, command sequence).

P4 designs a logging infrastructure for storing such immutable proofs of existence using DLT. To this end, the generic process of a non-repudiation service [1] is adopted to enumerate requirements for a DLT-based approach. The main challenges are ensuring adequate throughput in a decentralized network, and storing large volumes of log data. The resulting architecture stores log data off-chain to ensure availability and confidentiality, while storing proofs of existence (hash values) on a permissioned blockchain. The formal logging procedure follows three basic steps: *S1 Client application processing*, *S2 Blockchain node processing* and optionally *S3 Verification*. During *S1*, the log is received from the producing system and processed into a hash. That hash is then signed and packaged into a blockchain transaction. In *S2*, blockchain nodes store the hash along with a consensus-based timestamp. To prove a log entry's non-repudiation property retrospectively, *S3* may be used to verify its origin and integrity.

The formal procedure is implemented using an extended client-server architecture as shown in Figure 7. The client represents the user-facing frontend, while the server performs task *S1*. The transaction is submitted to a node of a public permissioned Exonum blockchain framework, which performs *S2* and *S3*.

The prototype is evaluated regarding security and performance. A threat analysis regarding fundamental distributed systems threats shows no major concerns, provided that operational security concerns are met before deployment (such as code audits, proper config-

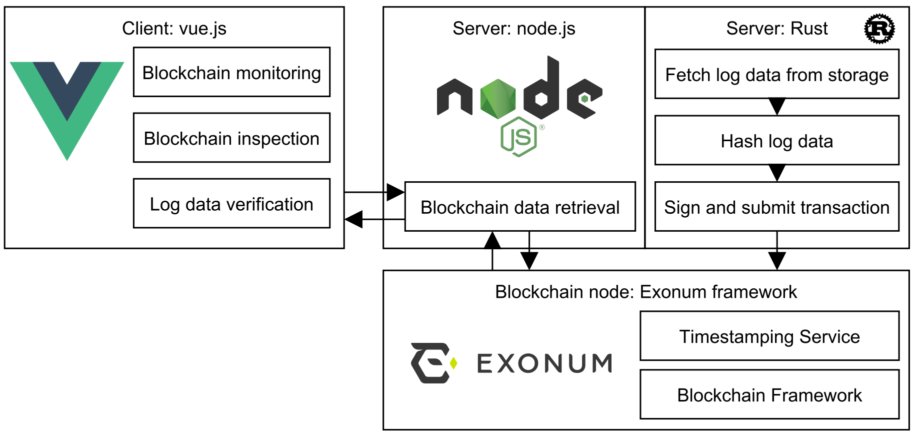


Figure 7: Architecture of the secure logging prototype.

uration). A performance benchmark demonstrates sufficient throughput of the blockchain network on the order of 3,000 transactions per second. Further scalability improvements can be achieved by batching entries.

Contribution of P4: The paper investigates the suitability of DLT for the purpose of secure logging. It goes beyond extant work by providing non-repudiable proofs of existence using a customized high-throughput and byzantine-fault tolerant blockchain network. A formal secure logging procedure and threat analysis ensure the non-repudiation of the proofs in audits by external stakeholders such as courts and insurance providers.

4.2 RQ2: Protecting DLT applications from threats

A significant amount of trust is placed into DLT applications due to their reputation for security. However, despite this reputation, successful attacks on DLT are widespread. This is most apparent in permissionless blockchains such as Ethereum. There, smart contract hacks are frequent with cryptocurrency thefts equivalent to millions of dollars [8]. Similar attacks apply to permissioned blockchains, even though they are rarely publicized due to their private nature. Besides smart contract threats, there are other types of attacks concerning identity management (e.g. credential theft) and networking (e.g. denial of service). In addition, in permissioned networks insiders from competing organizations on the network may behave maliciously. For example, if the network only uses a crash-fault tolerant consensus algorithm, it is ill-equipped to handle the intentional misbehavior of insider attacks. To manage these various threats and risks, organizations must know which threats DLT applications are facing (cf. **BP2-1**) and how to detect them (cf. **BP2-2**). This dissertation focuses on these aspects of DLT security management.

The contributions of this dissertation toward the aforementioned business problems are highlighted based on the NIST Cybersecurity Framework [6] shown in Figure 8. P5 focuses on the *Identify* and *Protect* areas by outlining threats and mitigation options. It focuses specifically on insider threats, which have received far less attention in research than external threats to DLT networks. P6 and P7 focus on the permissioned blockchain framework Hyperledger Fabric, which is the most frequently used platform for permissioned enterprise DLT applications [34]. Both publications enumerate and structure existing threats based on prior work, thus contributing to *Identify*. Their main contribution, however, lies in analyzing the feasibility of detecting threats, thereby addressing the *Detect* category. P6 examines the detectability of attacks using existing metrics. P7 uses the applicable metrics to build visualizations tailored for security experts to detect threats.

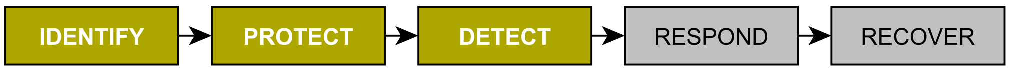


Figure 8: Aspects of the NIST Cybersecurity Framework covered by contributions.

4.2.1 Trust Factors and Insider Threats in Permissioned Distributed Ledgers - An analytical study and evaluation of popular DLT frameworks [P5]

DLT is often regarded as trust-free, while in fact there are many trusted components that are part of a typical DLT architecture. P5 focuses on this fact and analyzes the trust actors, layers and components that humans rely on when interacting with a DLT application. Trust relationships can be exploited by insiders for malicious attacks, which are explored in-depth in the second part of the publication.

The paper explores the known threat category of insider threats within the novel context of distributed ledgers. It structures the results of prior research and builds theory on trust

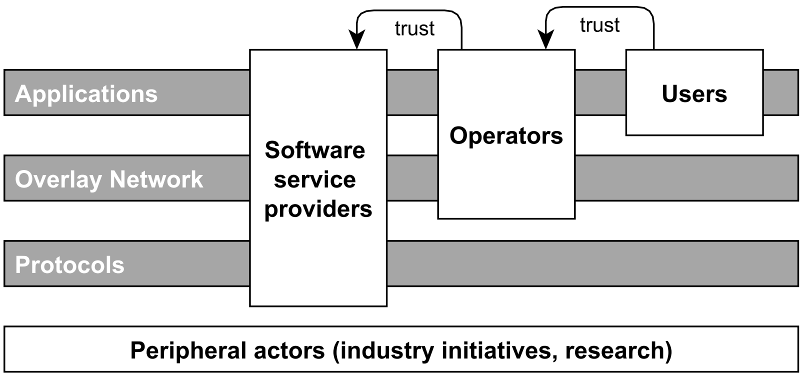

trust
trust
Applications
Overlay Network
Protocols
Software service providers
Operators
Users
Peripheral actors (industry initiatives, research)

Figure 9: Trust actors and layers within scope.

factors and insider threats by synthesizing extant literature. The theoretical model is evaluated through an analytical study on the technical components of widely used DLT frameworks.

The first part focuses on establishing a trust model for DLT. The model is shown in Figure 9. Three groups of actors are defined: *Users*, *Operators* and *Software Service Providers*, representing trustor-trustee relationships in this order. *Peripheral actors* such as industry initiatives (i.e. Hyperledger) and research institutions also contribute open-source trust components. These actors use or develop DLT software, which consists of three abstract trust layers. *Protocols* define the low-level storage, cryptography and network related tasks occurring behind the scenes. They are used by the *Overlay Network* to manage identity and establish consensus, where operators trust one another to behave honestly. Finally, applications with on- and off-chain components are built on top of this network, representing the user-facing abstraction layer.

Based on this trust model, P6 further examines how different trust actors may act as malicious insiders to exploit trust assumptions. Four types of outcomes may be achieved with an attack on specific data: *Modification*, *Destruction*, *Disclosure* and *Denial of Use*. Software service providers have extensive possibilities due to source code access and may achieve all of these consequences through vulnerability injection and abuse. Other trust actors are more limited in the scope of any single attack type. Operators may perform *Denial of Service*, *Data Manipulation*, *Credential Compromise* and *Malicious Misconfiguration* attacks. Users are limited to *Unauthorized Operations*. However, the design of DLT limits the impact of most attacks due to the built-in fault-tolerance, integrity preservation and replication characteristics. For example, insider threats by *Operators* and *Users* target only a single node or application. On the other hand, injected vulnerabilities can be considered one of the most significant threats, since all DLT nodes of the network are running the same or similar software versions. This could result in attacks that affect all nodes at the same time, potentially voiding the aforementioned safeguards and resulting in permanent disclosure, modification or destruction of data. This attack scenario is shown in Figure 10.

To build the insider threat model, each of the threats is mapped to the DLT components it affects. To provide guidance for practice, the components of four popular DLT frameworks are evaluated: Hyperledger Fabric, Hyperledger Sawtooth, go-ethereum and Corda. Each

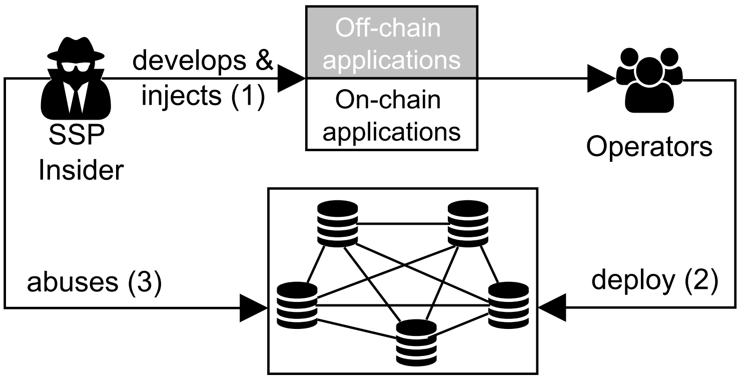


Figure 10: DLT vulnerability injection attack scenario.

framework's components are investigated for applicable insider threats based on the extant framework-specific literature. Finally, appropriate operational and technical mitigations are developed for each of the insider threats. On the operational side, legal agreements ensure liability for insider attacks. Code review and manual activity monitoring help detect malicious activity. Technical measures involve typical security practices such as identity and access management and timely software updates. In addition, automated detection of manipulation attempts and anomalies (i.e. using dedicated scanners) can help with detection.

Contribution of P6: P6 provides a detailed trust model for DLT frameworks, highlighting the relationships among trust actors and trusted DLT components. Based on the trust model, possible insider attacks for each trust actor are enumerated and categorized. These attack vectors are mapped to vulnerable DLT components in an analytical study of four popular DLT frameworks. Operational and technical mitigation options are highlighted for each attack vector category to provide guidance for practice.

4.2.2 Detecting Blockchain Security Threats [P6]

P6 focuses on threat detection for *permissioned* DLT. This subject has received far less attention by researchers than smart contract attack detection in *permissionless* DLT [8]. As detailed in P6, permissioned DLT systems are subject to various attacks unique to a permissioned environment, i.e. insider threats. These attacks require specialized monitoring systems designed for distributed networks. In addition, DLT nodes consist of multiple components (networking, consensus, execution) with different threats and monitoring requirements. P7 provides an overview of these threats and develops corresponding threat indicators. Due to its relevance in practice, the permissioned blockchain framework Hyperledger Fabric is chosen for evaluation. As part of the evaluation, the feasibility of measuring the threat indicators is examined based on the available data sources in Hyperledger Fabric.

To lay the groundwork for security monitoring in permissioned blockchains, P6 establishes a **threat model** of blockchain actors. The threat actors in the threat model are detailed through data flow diagrams. They indicate the relationship among blockchain components and blockchain actors. The threats themselves are categorized into vulnerabilities and attacks of malicious intent. Vulnerabilities may be due to misconfiguration, or caused by developers

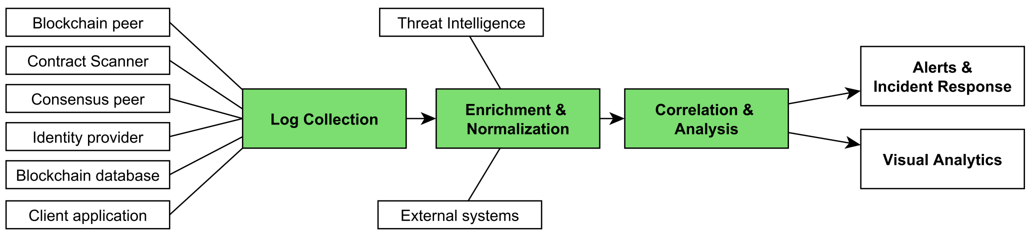


Figure 11: Monitoring architecture for detection of blockchain security threats.

in the form of vulnerable software. Malicious intent based threats vary in scope depending on the threat actor, which can be internal or external.

A systematic literature review is conducted to gather known threats in permissioned blockchains, yielding tens papers with relevant threats. Nine **attack categories** are determined by applying the threat model to the individual threats. Vulnerabilities are grouped into Contract, Framework, Dependency and Cryptographic Vulnerabilities, depending on the vulnerability type. Malicious Intent concerns Denial of Service, Network Partitioning, Consensus manipulation (behavior and configuration) and Identity Provider compromise. For each of these categories, one or multiple monitoring targets are defined for a total of 16 **threat indicators**. For example, Denial of Service threats can be detected through significant anomalies in transaction throughput/latency, incoming network messages and outstanding transactions. To collect and process the data needed for the threat indicators, a monitoring architecture is proposed, shown in Figure 11. It consists of the three steps *Log Collection*, *Enrichment & Normalization* and *Correlation & Analysis*.

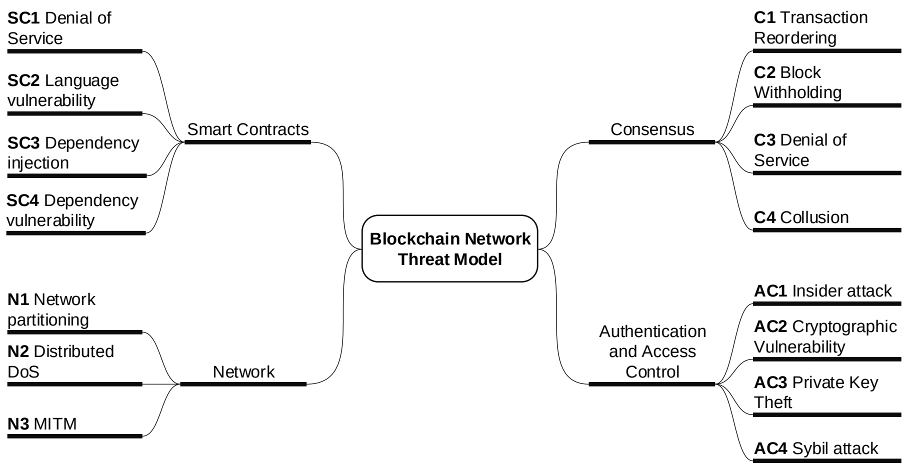
The 16 threat indicators are evaluated using Hyperledger Fabric data sources. For 11 indicators, sufficient data was available to detect relevant threats. For three, namely *Threat intelligence*, *Discarded blocks* and *Consensus leader election frequency*, data availability was limited. For *Scanned vulnerabilities* and *Outstanding transactions*, insufficient data was available. These results lead to suggestions for future research and proposed improvements for Hyperledger Fabric APIs.

Contribution of P6: P6 investigates threats for permissioned blockchains and their detection feasibility. A systematic literature review leads to an overview of existing threats, which are used to develop applicable threat indicators. To measure these indicators, a monitoring pipeline is proposed for aggregating and processing the applicable data sources.

4.2.3 HyperSec: A Visual Analytics approach to blockchain monitoring [P7]

P6 examined the feasibility of detecting attacks and determined available data sources in Hyperledger Fabric. P7 builds on these results and goes one step further. It develops a task-appropriate visualization assistance for DLT security experts to support the practical detection of threats. Prior to P8, extant blockchain security research mainly focused smart contract and threat research. This work goes beyond existing work on automated threat detection by integrating human security experts as part of DLT threat monitoring. In practice, humans are always involved in the incident response process [60], hence task support tools are a necessary step toward practical threat monitoring for DLT.

To develop an appropriate task support tool, the paper follows the DSRM [48] by defining the problem, objective and requirements. Subsequently, an artifact is designed and developed (task-oriented visualizations). The artifact is then implemented in a prototype for evaluation. As the design methodology, P7 uses the nested blocks and guidelines model (NBGM) [43] to develop comprehensive and task-oriented visualizations. The NBGM is based on the definition of the domain problem (blockchain security monitoring) and derives intended users, their tasks and applicable data elements. Based on these building blocks, design requirements are defined for a prototype to support the users' tasks with data visualizations.



SC1 Denial of Service
SC2 Language vulnerability
SC3 Dependency injection
SC4 Dependency vulnerability
Smart Contracts
N1 Network partitioning
N2 Distributed DoS
N3 MITM
Network
Blockchain Network Threat Model
Consensus
C1 Transaction Reordering
C2 Block Withholding
C3 Denial of Service
C4 Collusion
Authentication and Access Control
AC1 Insider attack
AC2 Cryptographic Vulnerability
AC3 Private Key Theft
AC4 Sybil attack

Figure 12: Attack tree of permissioned blockchain threats.

First, the domain problem is characterized based on the detection process (see Figure 8). P7 focuses on the Detect function, which consists of the four steps *Collection*, *Aggregation*, *Visualization* and *Analysis*. The intended users of the framework are the domain experts, i.e. security professionals with knowledge about blockchain threats. Their tasks are defined based on relevant threats. Threats are determined based on the results of P5, P6 and related literature surveys. The result is shown in attack tree form in Figure 12. An overview of tasks for detecting these threats is given in Table 3. Each task is suitable to detect several types of threats. In the event of an attack, domain experts intuitively combine these tasks during an in-depth investigation.

Five design requirements are determined based on the domain problem and objective of the design. Users should be able to view *General Security Information (R1)*. This refers to configuration changes and known vulnerabilities (*T1 ,T2, T7*). A *Network View (R2)* provides information about network activities and metrics (*T4, T8*). To view block and transaction activity, a dedicated *Transaction View (R3)* is needed (*T5, T6*). The mentioned views need to support *Interactivity and Details(R4)* to permit in-depth investigations, i.e. by viewing log files (*T3*).

Based on these requirements, a prototype is developed termed HyperSec (**Hyper**ledger

Table 3: Security expert tasks and related attacks (adapted from P7).

Task	Description	Related attacks
T1	Identify vulnerable smart contracts	SC1, SC2, SC3
T2	Identify blockchain framework vulnerabilities	SC4, AC2
T3	Inspect log files of running services on demand	SC4, N1, N3, C3, C4
T4	Review networking activity	N1, N2, N3
T5	Compare transaction metrics over time	N2, C2
T6	Explore block and transaction history	SC1, SC2, C3, AC1
T7	Review configuration changes	C1, AC1
T8	Detect identity abuse	AC1, AC3, AC4

Security Explorer). The prototype builds on the open-source Hyperledger Explorer, a general blockchain explorer to view blockchain data. It is enhanced with additional data sources and views to support the design requirements. The resulting architecture is shown in Figure 13. To support the required tasks, the backend is enhanced with Hyperledger Fabric metrics sourced through Prometheus and Docker logs from the Docker API.

Regarding the frontend, a new Chaincodes view is added and vulnerability information is shown on the *Dashboard* view to fulfill R1. The *Network* view is enhanced with a node-link diagram of the network showing current metrics in fulfillment of R2. The *Transactions* view is enhanced with interactive transaction activity charts to support R3. R4 is built into each of these three views through interactivity and detailed data inspection features (transaction/log search and detail view).

Finally, three attacks are simulated on a test network and evaluated using the HyperSec prototype: a chaincode vulnerability (*SC2*), a distributed denial of service attack (*N2*) and insider credential abuse (*AC1*). Each attack can be clearly distinguished from regular activity using the enhanced visualization and inspection capabilities of HyperSec.

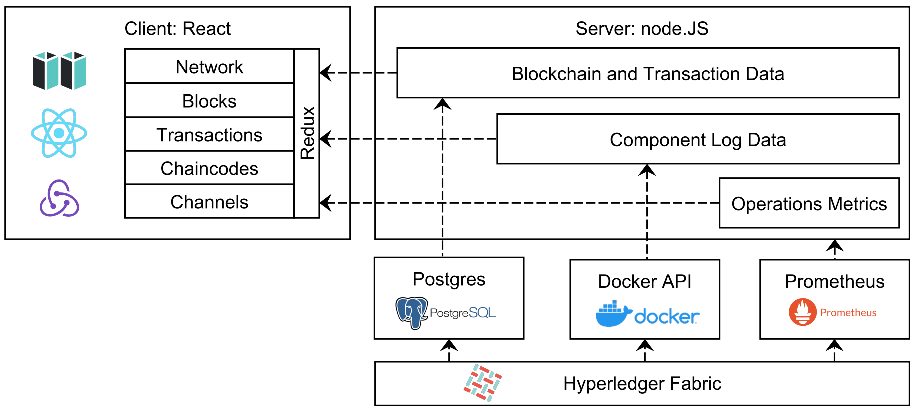


Figure 13: Architecture of the HyperSec prototype.

Contribution of P7: P7 contributes a task-centered design and prototype to detect blockchain security threats. The design consists of interactive visualizations based on adequate blockchain data sources. It is tailored towards the tasks necessary for security monitoring, carried out by domain experts. P7 includes a prototypical implementation of the design targeting the blockchain framework Hyperledger Fabric. The prototype demonstrates the feasibility of detecting attacks through the structured, comprehensive and concise aggregation of data in tables and visualizations. An evaluation is performed for three specific attacks, highlighting the relevant tasks and views in the prototype.

In addition to the conference presentation, this paper was presented as a technical talk at the industry conference *Hyperledger Global Forum 2021*.

4.3 RQ3: Secure Information Sharing in a DLT Consortium

The institutionalized form of a collaboration with the goal of sharing resources and know-how among businesses is referred to as a consortium [71]. Collaboration in a consortium is non-trivial and requires the partners to navigate conflicting needs and tensions [72]. These tensions can be the result of competitive behavior or merely opportunism in an otherwise beneficial collaboration [44]. Eventually, these organizational tensions may evolve into severe technical security issues such as insider threats, which were pointed out in Section 4.2. To prevent escalating tensions and ensure healthy collaboration, it is vital for consortium partners to establish a solid foundation for their partnership. A key part of this process is establishing governance mechanisms, legal foundations and a joint vision. Hence, to prevent the development of security threats from internal partners, this dissertation investigates contributing factors toward successful and secure consortium collaboration (**BP3-1**). Conclusively, the development of collaboration principles, standards and guidelines also contributes toward the overall goal of fostering DLT adoption.

The topic is addressed in publication P8 as part of a multiple case study. While this methodology departs from the DSRM used in most other works in this dissertation, it is necessary to study these behavioral organizational phenomena. Based on empirical work with literature and expert interviews, the publication derives the foundations of existing successful interorganizational collaborations. A lifecycle provides a guideline for developing consortia towards fruitful evolution and expansion.

4.3.1 Comparing Successful DLT Consortia: A Lifecycle Perspective [P8]

P8 considers DLT consortia from a strategic development perspective. It researches successful DLT consortia to find commonalities and differences. The overall goal is the development of guidance for newer consortia. Different to previous publications in this dissertation, it does not use design science research. Instead, it relies on case study methodology to find and analyze relevant cases. Case studies are well suited to study contemporary phenomena in a real-life context [70] and thus ideal to investigate current DLT consortium development. Specifically, a holistic multiple-case design is chosen to compare several consortia. A comparison of these consortia is used to determine a common lifecycle. The case study data and lifecycle are evaluated using expert interviews with representatives of the selected consortia.

The paper is structured along the two research questions, which are addressed in separate sections after introducing the methodology and selected cases. The first research question addresses commonalities and differences between operational DLT consortia. The second analyzes the different phases which DLT consortia undergo during their lifecycle.

First, successful consortia are identified based on the Information Quality (IQ) framework. Intrinsic and contextual IQ are used to filter relevant data from the wealth of information about DLT consortia in gray and white literature. The filtering criteria focus on information availability, consortium age and the number of organizations operating DLT nodes. In total, 33 consortia were found and analyzed. After filtering, a total of nine cases remained as subjects of study. They are shown in Table 4.

Table 4: Overview of selected case studies (in alphabetical order).

Name	Lead	Partner	Platform	Sector	Nodes
B3i Re	B3i Services	R3	Corda	Insurance	21
Bakong	NBC	Soramitsu	Iroha	CBDC	16
Cardossier	cardossier	AdNovum	Corda	Vehicles	8
Contour	Contour	Contour	Corda	Trade Finance	8
covantis	Covantis	ConsenSys	Quorum	Agriculture	18
DL Freight	Walmart Canada	DLT Labs	Fabric	Logistics	30
MediLedger	Chronicled	Chronicled	Ethereum	Pharma	10
TradeLens	Maersk	IBM	Fabric	Logistics	14
we.trade	we.trade	IBM	Fabric	Trade Finance	16

For each consortium, all available white and gray literature is aggregated and filtered for quality. The information obtained from data collection is used to prepare key findings for expert interviews. As part of the interviews, the consortium representatives were asked to provide their opinion on the findings and an initial version of a consortium lifecycle. The feedback is then used to further improve the accuracy of the results, which are described in the following.

Six comparison dimensions are derived from blockchain governance literature: *Platform choice*, *Network size*, *Incentives*, *Legal Form*, *Disintermediation* and *Interoperability*. The results showed Hyperledger Fabric and R3 Corda as the most common platforms of choice with network size varying between 8 and 30 operators. The main incentive for businesses to join were the business benefits. The legal form was less clear, albeit most consortia founded a separate entity to handle platform development. Interoperability was of growing concern in all consortia, but only few had implemented prototypes at the time of study.

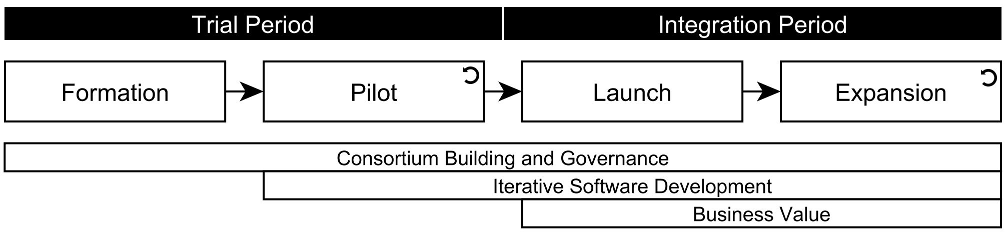


Figure 14: DLT consortium lifecycle phases.

Research for the 9 selected consortia yielded several common milestones during their development: the *Foundation*, *Institutionalization*, *Pilot* and *Launch*. Based on these milestones, four lifecycle phases for DLT consortia are developed (shown in Figure 14). The *Formation* phase involves first software prototypes, a search for partners, and optionally institutionalization if the partners decide to incorporate a separate entity. During the following *Pilot* phase, requirements for the target business case are settled, along with development and testing of a prototype for the pilot experiment. Depending on the success of the pilot, multiple pilots may be carried out. Eventually, consortia proceed to the *Launch* phase, with successful commercialization of the software product and first business value being delivered. Subsequently, during the *Expansion* phase the scope of the product is increased to include other business cases. Additional partners are added to the ecosystem beyond the main consortium shareholders to improve integration and transparency throughout the business network.

Beyond the development lifecycle, two characteristics were observed during the case study. First, consortium management tends to fall either in the category of *democratic governance* or *benevolent dictatorship*. Benevolent dictatorship was the case when there was a dominant actor within the supply chain (i.e. Walmart for DL Freight). Consortia with organizations of comparable size (i.e. Cardossier) tended to behave more democratic.

Contribution of P8: P8 provides collaboration and development guidelines for new and existing blockchain consortia. The guidelines are based on empirical case study and interview data. Specifically, commonalities and differences among successful consortia are highlighted. A consortium lifecycle is proposed based on common milestones, and all results are verified in expert interviews. The lifecycle is intended to foster successful consortium development and thereby general adoption of DLT.
This publication was invited for submission of an extended version to the Electronic Markets special issue on *"Enterprise and organizational applications of distributed ledger technologies"* during the HICSS conference presentation session.

4.4 Complementary publications

Besides the main publications P1 through P8, two additional publications emerged as part of the dissertation, shown in Table 5. These publications can be seen as complementary, since they do not answer the research questions directly, but contribute towards improving security in the DLT ecosystem.

Table 5: Overview of complementary publications.

No.	Title	Venue
A1	SSIBAC: Self-Sovereign Identity Based Access Control	The 3rd International Workshop on Blockchain Systems and Applications (BlockchainSys2020), in Conjunction with IEEE TrustCom 2020
A2	BISCUIT - Blockchain Security Incident Reporting based on Human Observations	not yet published

Paper A1

Title:	SSIBAC: Self-Sovereign Identity Based Access Control
Status:	published
Publication:	The 3rd International Workshop on Blockchain Systems and Applications (BlockchainSys2020), in Conjunction with IEEE TrustCom 2020
Submitted:	28 July 2020
Accepted:	14 October 2020
Citation:	Rafael Belchior, Benedikt Putz, Günther Pernul, Miguel Correia, André Vasconcelos, and Sérgio Guerreiro. 2020. SSIBAC: Self-Sovereign Identity Based Access Control. In The 3rd International Workshop on Blockchain Systems and Applications (BlockchainSys2020), in Conjunction with IEEE TrustCom 2020, IEEE.

Paper A1 explores the concept of Self-Sovereign Identity (SSI). SSI represents a novel decentralized identity approach, which enhances user privacy and control. It is realized through decentralized identifiers (DIDs) and verifiable credentials (VCs). Issuers provide these VCs to users, who can then present them to Verifiers to prove some fact, for example that the age is within a specific range. These credentials are comparable to attributes in traditional attribute-based access control (ABAC). Paper A1 investigates if VCs can be used to implement an access control model for cross-organizational identity management. The research result is the SSIBAC prototype based on Hyperledger Aries, which demonstrates that access control policies can be enforced based on VCs. Aries relies on the blockchain framework Hyperledger Indy as its Verifiable Credential Registry. A technical evaluation demonstrates good throughput and latency for credential generation/revocation and access control requests.

Research work on Paper A1 was conducted jointly with the Instituto Superior Técnico in

Lisbon, Portugal. It was published at the 3rd International Workshop on Blockchain Systems and Applications (BlockchainSys2020), held in conjunction with IEEE TrustCom 2020.

Paper A2

Title:	BISCUIT - Blockchain Security Incident Reporting based on Human Observations
Status:	under review
Submitted:	23 March 2022
Citation:	Benedikt Putz, Manfred Vielberth, and Günther Pernul. 2022. BISCUIT - Blockchain Security Incident Reporting based on Human Observations. Working Paper, University of Regensburg, 2022.

Paper A2 investigates human aspects of blockchain security. Its main goal is to improve the incident response process by integrating human observations. The presented approach enables structured security incident reporting by security novices and subsequent incident discussion by security experts. To enumerate and structure existing threats, the paper first develops a taxonomy of blockchain security threats based on a literature review. The taxonomy is then used to build an interactive form to report threat sources, events and affected entities. The form is part of the frontend of a larger prototype called BISCUIT. The prototype also includes processing and storage backends to support the reporting process. A Python API processes incidents and matches them with previously reported similar incidents to assist users in filling out the form. The reported incident is then kept in local storage until it is approved by a security expert. On approval, incident metadata is stored on the blockchain to preserve its integrity and to enable decentralized discussion. Full incident data is stored off-chain in a distributed hash table. A security committee consisting of multiple security experts discusses on-chain incidents using a comment-based discussion system to reach a decision on incident response. A user study evaluates the usability of the user-facing frontend and the contained taxonomy.

5 Conclusion and future work

The overall objective of this dissertation was the development of DLT applications for secure interorganizational information sharing. Applications are needed to bring the various benefits of DLT for organizations to practice contexts - i.e. transparency, non-repudiation, auditability of shared data. Finding such use cases is not trivial, as specific prerequisites must be met for DLT to be a reasonable and cost-effective fit [65, 40].

To meet this objective, RQ1 focuses on developing such novel use cases of DLT. Two sectors are identified that presented research opportunities for DLT applications: the Industrial Internet of Things (IIoT) and Cybersecurity. Within the IIoT, the Digital Twin paradigm is found to be well suited for development as an information sharing artifact using DLT. In Cybersecurity information sharing, two use cases are identified: the sharing of non-repudiation proofs for auditable logging and the sharing of threat intelligence.

RQ2 focuses on protecting these applications from threats. Hence, the first work to address this question identifies trusted actors and components specific to the permissioned interorganizational environment. Research results include a list of insider threats and potential mitigations. Subsequent research works address the detection of these internal threats and known external threats. These are first aggregated in a threat model and taxonomy, before determining metrics to detect them on the permissioned DLT platform Hyperledger Fabric. Based on these results, DLT threat monitoring is further developed by designing custom visualizations for threat detection, to be used by DLT security experts.

The final research question RQ3 tackles the topic of consortium collaboration. The management of interorganizational tensions and the temporal development of consortia are studied in a comparative multiple case study. The results of a literature review are verified in expert interviews, yielding an overview of the development of nine successful DLT consortia. Based on the milestones of their development, a lifecycle for DLT consortium development is abstracted and elaborated in detail. This lifecycle and the lessons learned by these consortia serve as guidance for new and developing consortia. The intent is to permit avoidance of tensions and to increase the security of shared information through mutual trust. While DLT shifts some trust towards cryptographic assurances, increased trust is nevertheless beneficial to avoid any insider threats.

In summary, this dissertation advances the state of the art in DLT research by addressing various research gaps. It develops novel applications for organizational use cases in the IIoT and Cybersecurity domains. Furthermore, it establishes foundations for the secure adoption of DLT in organizations by studying DLT threats and developing methods for detecting attacks. Beyond technical security, it also studies and derives guidelines for the establishment of sound and successful DLT consortia.

There are several directions for future research enabled by the works of this dissertation. Many avenues are mentioned in the individual works, but hereafter we give a high-level perspective. To improve **DLT-based digital twins**, researchers should investigate enhanced functionality and performance artifacts to enhance their usability in practice. For example, data analytics functionality and compensation could become part of the Digital Twin. Regarding scalability, integration in an industrial environment can demonstrate the practical feasibility and constraints of handling large amounts of sensor data. In **Cybersecurity information sharing**, threat intelligence sharing using DLT has emerged as a promising area of research. While P4 demonstrated the feasibility of sharing critical infrastructure CTI, the design space of possible solutions is not yet exhausted. Researchers should investigate other DLT frameworks and data sharing protocols to find appropriate solutions for all types of CTI sharing contexts. **Protecting DLT from internal and external threats** remains especially relevant with the emergence of interoperability among distributed ledgers. The security of interoperability constructs among multiple permissioned and among permissioned and permissionless networks remains unclear. Threats to individual ledgers have received significant amounts of attention, but most research has focused on Ethereum and Hyperledger Fabric, resulting in a need to study other DLT frameworks. Beyond threat identification, this dissertation has focused in part on **threat detection**. The next step in the response process is DLT incident mitigation, which provides avenues for research in both the permissioned and permissionless environments. To enable **secure DLT consortium collaboration**, this work has established a lifecycle for consortium development. Future work should investigate not just successful consortia but also failed or troubled consortia to determine antecedents of failure and interorganizational tensions. In addition, empirical research and theory development is needed regarding decision paths for consortium mergers and consortium interoperability.

Overall, this dissertation develops a sound basis for future research, enabling both incremental improvements and novel research directions.

Bibliography

[1] Accorsi, R. Log data as digital evidence: What secure logging protocols have to offer? *Proc. of the International Computer Software and Applications Conference 2* (2009), 398–403.

[2] Alammary, A., Alhazmi, S., Almasri, M., and Gillani, S. Blockchain-based applications in education: A systematic review. *Applied Sciences 9*, 12 (2019).

[3] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S. W., and Yellick, J. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proceedings of the Thirteenth EuroSys Conference* (New York, NY, USA, 2018), ACM, pp. 30:1–30:15.

[4] Ayman, A., Aziz, A., Alipour, A., and Laszka, A. Smart Contract Development in Practice: Trends, Issues, and Discussions on Stack Overflow. *CoRR abs/1905.0* (2019).

[5] Berdik, D., Otoum, S., Schmidt, N., Porter, D., and Jararweh, Y. A Survey on Blockchain for Information Systems Management and Security. *Information Processing & Management 58*, 1 (Jan. 2021).

[6] Calder, A. *NIST Cybersecurity Framework.* 2018.

[7] Casino, F., Dasaklis, T. K., and Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics* (2019).

[8] Chen, H., Pendleton, M., Njilla, L., and Xu, S. A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. *ACM Comput. Surv. 53*, 3 (2020), 67:1–67:43.

[9] Chen, T., Cao, R., Li, T., Luo, X., Gu, G., Zhang, Y., Liao, Z., Zhu, H., Chen, G., He, Z., Tang, Y., Lin, X., and Zhang, X. SODA: A Generic Online Detection Framework for Smart Contracts. In *27th Annual Network and Distributed System Security Symposium, {NDSS} 2020* (2020), The Internet Society.

[10] COYNE, E. J., AND WEIL, T. R. ABAC and RBAC: Scalable, Flexible, and Auditable Access Management. *IT Professional 15*, 3 (2013), 14–16.

[11] DABHOLKAR, A., AND SARASWAT, V. Ripping the Fabric: Attacks and Mitigations on Hyperledger Fabric. In *Applications and Techniques in Information Security - 10th International Conference, ATIS 2019, Thanjavur, India, November 22-24, 2019, Proceedings* (2019), V. S. S. Sriram, V. Subramaniyaswamy, N. Sasikaladevi, L. Zhang, L. Batten, and G. Li, Eds., vol. 1116 of *Communications in Computer and Information Science*, Springer, pp. 300–311.

[12] DAVENPORT, A., SHETTY, S., AND LIANG, X. Attack Surface Analysis of Permissioned Blockchain Platforms for Smart Cities. In *2018 IEEE International Smart Cities Conference, ISC2 2018* (2019).

[13] DE AGUIAR, E. J., FAIÇAL, B. S., KRISHNAMACHARI, B., AND UEYAMA, J. A survey of blockchain-based strategies for healthcare. *Acm Computing Surveys 53*, 2 (Mar. 2020).

[14] DIETZ, M., AND PERNUL, G. Digital Twin: Empowering Enterprises Towards a System-of-Systems Approach. *Business & Information Systems Engineering 62*, 2 (2020), 179–184.

[15] DZIEMBOWSKI, S., ECKEY, L., AND FAUST, S. FairSwap: How To Fairly Exchange Digital Goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018* (2018), D. Lie, M. Mannan, M. Backes, and X. Wang, Eds., ACM, pp. 967–984.

[16] ENISA. ENISA Threat Landscape - Top 15 Threats. Tech. rep., ENISA, 2019.

[17] FERGUSON, M., AND KETZENBERG, M. E. Information Sharing to Improve Retail Product Freshness of Perishables. *Production and Operations Management* (2006).

[18] FIALA, P. Information sharing in supply chains. *Omega* (2005).

[19] GREENSPAN, G. MultiChain Private Blockchain - White Paper. 1–17.

[20] GREGOR, S., AND HEVNER, A. R. Positioning and Presenting Design Science Research for Maximum Impact. *MIS Q. 37*, 2 (2013), 337–355.

[21] GUGGENBERGER, T., SCHWEIZER, A., AND URBACH, N. Improving Interorganizational Information Sharing for Vendor Managed Inventory: Toward a Decentralized Information Hub Using Blockchain Technology. *IEEE Transactions on Engineering Management* (2020).

[22] HEARN, M. Corda: A distributed ledger. *Corda Technical White Paper 2016* (2016).

[23] HENNINGSEN, S. A., TEUNIS, D., FLORIAN, M., AND SCHEUERMANN, B. Eclipsing Ethereum Peers with False Friends. In *2019 IEEE European Symposium on Security*

and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019 (2019), IEEE, pp. 300–309.

[24] HEVNER, A. R., MARCH, S. T., PARK, J., AND RAM, S. Design science in information systems research. *MIS Quarterly: Management Information Systems* (2004).

[25] HOMOLIAK, I., VENUGOPALAN, S., REIJSBERGEN, D., HUM, Q., SCHUMI, R., AND SZALACHOWSKI, P. The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. *IEEE Communications Surveys and Tutorials* (2020).

[26] HUANG, H., KONG, W., ZHOU, S., ZHENG, Z., AND GUO, S. A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools. *ACM Comput. Surv. 54*, 2 (2021), 44:1–44:42.

[27] HUANG, Y., BIAN, Y., LI, R., ZHAO, J. L., AND SHI, P. Smart Contract Security: A Software Lifecycle Perspective. *IEEE Access 7* (2019), 150184–150202.

[28] IBM. We.trade | IBM. https://www.ibm.com/case-studies/wetrade-blockchain-fintech-trade-finance, 2020.

[29] IVANOV, N., LOU, J., CHEN, T., LI, J., AND YAN, Q. Targeting the Weakest Link: Social Engineering Attacks in Ethereum Smart Contracts. In *ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021* (2021), J. Cao, M. H. Au, Z. Lin, and M. Yung, Eds., ACM, pp. 787–801.

[30] JENSEN, T., HEDMAN, J., AND HENNINGSSON, S. How TradeLens delivers business value with blockchain technology. *MIS Quarterly Executive* (2019).

[31] JP MORGAN CHASE. Quorum Whitepaper. Tech. rep., 2016.

[32] KACHERGINSKY, P. Blockchain Threat Intelligence | Peter Kacherginsky | Substack. https://www.blockthreat.io/, 2022.

[33] KIAYIAS, A., RUSSELL, A., DAVID, B., AND OLIYNYKOV, R. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I* (2017), J. Katz and H. Shacham, Eds., vol. 10401 of *Lecture Notes in Computer Science*, Springer, pp. 357–388.

[34] KOLB, J., ABDELBAKY, M., KATZ, R. H., AND CULLER, D. E. Core Concepts, Challenges, and Future Directions in Blockchain: A Centralized Tutorial. *ACM Computing Surveys 53*, 1 (Feb. 2020), 9:1–9:39.

[35] KUMAR, R. S., AND PUGAZHENDHI, S. Information sharing in supply chains: An overview. In *Procedia Engineering* (2012).

[36] LACITY, M., AND VAN HOEK, R. What we've learned so far about blockchain for business. *MIT Sloan Management Review 62*, 3 (2021), 48–54.

[37] LARSON, A. Partner networks: Leveraging external ties to improve entrepreneurial performance. *Journal of business venturing 6*, 3 (1991), 173–188.

[38] LI, K., CHEN, J., LIU, X., TANG, Y. R., WANG, X., AND LUO, X. As Strong As Its Weakest Link: How to Break Blockchain DApps at RPC Service. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, Virtually, February 21-25, 2021* (2021), The Internet Society.

[39] LIPNER, S. Security development lifecycle. *Datenschutz und Datensicherheit-DuD 34*, 3 (2010), 135–137.

[40] MEIKLEJOHN, S. Top Ten Obstacles along Distributed Ledgers Path to Adoption. *IEEE Security Privacy 16*, 4 (July 2018), 13–19.

[41] MENGES, F., AND PERNUL, G. A comparative analysis of incident reporting formats. *Computers & Security 73* (2018), 87–101.

[42] MEYER, E., WELPE, I. M., AND SANDNER, P. G. Decentralized Finance—A Systematic Literature Review and Research Directions. SSRN Scholarly Paper 4016497, Social Science Research Network, Rochester, NY, Nov. 2021.

[43] MEYER, M., SEDLMAIR, M., QUINAN, P. S., AND MUNZNER, T. The nested blocks and guidelines model. *Information Visualization 14*, 3 (2015), 234–249.

[44] MISHRA, D. P., KUKREJA, R. K., AND MISHRA, A. S. Blockchain as a governance mechanism for tackling dark side effects in interorganizational relationships. *International Journal of Organizational Analysis 30*, 2 (Jan. 2021), 340–364.

[45] NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 9.

[46] NGUYEN, T. D., PHAM, L. H., AND SUN, J. SGUARD: Towards Fixing Vulnerable Smart Contracts Automatically. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021* (2021), IEEE, pp. 1215–1229.

[47] PEDERSEN, A. B., RISIUS, M., AND BECK, R. A ten-step decision path to determine when to use blockchain technologies. *MIS Quarterly Executive 18*, 2 (2019), 99–115.

[48] PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M. A., AND CHATTERJEE, S. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems 24*, 3 (2007), 45–77.

[49] POPOV, S. The Tangle, IOTA Whitepaper. 1–28.

[50] RAMEDER, H., DI ANGELO, M., AND SALZER, G. Review of Automated Vulnerability Analysis of Smart Contracts on Ethereum. *Frontiers in Blockchain 5* (2022).

[51] RAUCHS, M., BLANDIN, A., BEAR, K., AND MCKEON, S. B. 2nd Global Enterprise blockchain benchmarking study. *Available at SSRN 3461765* (2019).

[52] REYNA, A., MARTÍN, C., CHEN, J., SOLER, E., AND DÍAZ, M. On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems 88* (Nov. 2018), 173–190.

[53] RISIUS, M., AND SPOHRER, K. A Blockchain Research Framework. *Business & Information Systems Engineering 59*, 6 (2017), 385–409.

[54] SALMAN, T., ZOLANVARI, M., ERBAD, A., JAIN, R., AND SAMAKA, M. Security Services Using Blockchains: A State of the Art Survey. *IEEE Communications Surveys Tutorials 21*, 1 (2019), 858–880.

[55] STIFTER, N., JUDMAYER, A., AND WEIPPL, E. Revisiting Practical Byzantine Fault Tolerance Through Blockchain Technologies. In *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*, S. Biffl, M. Eckhart, A. Lüder, and E. Weippl, Eds. Springer International Publishing, Cham, 2019, pp. 471–495.

[56] SUHAIL, S., HUSSAIN, R., JURDAK, R., ORACEVIC, A., SALAH, K., HONG, C. S., AND MATULEVIČIUS, R. Blockchain-based Digital Twins: Research Trends, Issues, and Future Challenges. *ACM Computing Surveys* (Feb. 2022).

[57] TONG, P. Y., AND CROSNO, J. L. Are information asymmetry and sharing good, bad, or context dependent? A meta-analytic review. *Industrial Marketing Management* (2016).

[58] TRIENEKENS, J. H., WOGNUM, P. M., BEULENS, A. J., AND VAN DER VORST, J. G. Transparency in complex dynamic food supply chains. *Advanced Engineering Informatics* (2012).

[59] TSANKOV, P., DAN, A. M., DRACHSLER-COHEN, D., GERVAIS, A., BÜNZLI, F., AND VECHEV, M. T. Securify: Practical Security Analysis of Smart Contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018* (2018), D. Lie, M. Mannan, M. Backes, and X. Wang, Eds., ACM, pp. 67–82.

[60] VIELBERTH, M., BÖHM, F., FICHTINGER, I., AND PERNUL, G. Security Operations Center: A Systematic Study and Open Challenges. *IEEE Access 8* (2020), 227756–227779.

[61] WAN, Z., XIA, X., LO, D., CHEN, J., LUO, X., AND YANG, X. Smart Contract Security: A Practitioners' Perspective. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (2021), IEEE, pp. 1410–1422.

[62] WÖHRER, M., AND ZDUN, U. Smart contracts: Security patterns in the ethereum ecosystem and solidity. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (Mar. 2018), pp. 2–8.

[63] WÖHRER, M., ZDUN, U., AND RINDERLE-MA, S. Architecture Design of Blockchain-Based Applications. In *2021 3rd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)* (Sept. 2021), pp. 173–180.

[64] WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* (2014), 1–32.

[65] WÜST, K., AND GERVAIS, A. Do you Need a Blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (2018), pp. 45–54.

[66] XU, X., PAUTASSO, C., ZHU, L., GRAMOLI, V., PONOMAREV, A., TRAN, A. B., AND CHEN, S. The blockchain as a software connector. In *Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016* (Apr. 2016), IEEE, pp. 182–191.

[67] XU, X., PAUTASSO, C., ZHU, L., LU, Q., AND WEBER, I. A pattern collection for blockchain-based applications. In *ACM International Conference Proceeding Series* (2018).

[68] XU, X., WEBER, I., AND STAPLES, M. *Architecture for Blockchain Applications*. 2019.

[69] YAMASHITA, K., NOMURA, Y., ZHOU, E., PI, B., AND JUN, S. Potential Risks of Hyperledger Fabric Smart Contracts. In *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (2019), pp. 1–10.

[70] YIN, R. K. *Case Study Research: Design and Methods*. 2009.

[71] ZAVOLOKINA, L., ZIOLKOWSKI, R., BAUER, I., AND SCHWABE, G. Management, governance, and value creation in a blockchain consortium. *MIS Quarterly Executive* (2020).

[72] ZIOLKOWSKI, R., MISCIONE, G., AND SCHWABE, G. Decision Problems in Blockchain Governance: Old Wine in New Bottles or Walking in Someone Else's Shoes? *Journal of Management Information Systems* (2020).

[73] ZIOLKOWSKI, R., PARANGI, G., MISCIONE, G., AND SCHWABE, G. Examining gentle rivalry: Decision-making in blockchain systems. In *52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019* (2019), T. Bui, Ed., ScholarSpace, pp. 1–10.

Part II

Research papers

The second part of this dissertation contains publications P1 through P8, with a brief introductory metadata summary of each publication followed by the full text.

1 RQ1: Novel DLT applications for secure information sharing

1.1 A distributed ledger approach to digital twin secure data sharing [P1]

Status:	published
Publication:	Data and Applications Security and Privacy XXXIII (DBSec'19)
Submitted:	11 March 2019
Accepted:	14 April 2019
Citation:	Marietheres Dietz, Benedikt Putz, and Günther Pernul. 2019. A Distributed Ledger Approach to Digital Twin Secure Data Sharing. In Data and Applications Security and Privacy XXXIII, Simon N. Foley (ed.). Springer International Publishing, 281–300.

Conference Description: DBSec is an annual international conference covering research in data and applications security and privacy. It is affiliated with the IFIP WG 11.3 Working Group on Data and Applications Security and Privacy.

A Distributed Ledger Approach to Digital Twin Secure Data Sharing

Marietheres Dietz(✉), Benedikt Putz, and Günther Pernul

University of Regensburg, Regensburg, Germany
{marietheres.dietz,benedikt.putz,guenther.pernul}@ur.de

Abstract. The Digital Twin refers to a digital representation of any real-world counterpart allowing its management (from simple monitoring to autonomy). At the core of the concept lies the inclusion of the entire asset lifecycle. To enable all lifecycle parties to partake, the Digital Twin should provide a sharable data base. Thereby, integrity and confidentiality issues are pressing, turning security into a major requirement. However, given that the Digital Twin paradigm is still at an early stage, most works do not consider security yet. Distributed ledgers provide a novel technology for multi-party data sharing that emphasizes security features such as integrity. For this reason, we examine the applicability of distributed ledgers to secure Digital Twin data sharing. We contribute to current literature by identifying requirements for Digital Twin data sharing in order to overcome current infrastructural challenges. We furthermore propose a framework for secure Digital Twin data sharing based on Distributed Ledger Technology. A conclusive use case demonstrates requirements fulfillment and is followed by a critical discussion proposing avenues for future work.

Keywords: Trust frameworks · Distributed systems security · Distributed ledger technology · Digital twin

1 Introduction

Hardly anything has revolutionized society as much as digitization. At its beginning, data from everyday life was captured and stored digitally. After reaching significant amounts of digital data, recent years have been devoted to gaining relevant insights into data by leveraging Big Data Analytics, Artificial Intelligence and so on. A next step in digitization is now emerging in the form of the Digital Twin (DT) paradigm.

The Digital Twin refers to a digital representation of any real-world counterpart, at most times an enterprise asset. Its core building blocks are asset-specific data items, often enhanced with semantic technologies and analysis/simulation

The first two authors have contributed equally to this manuscript.

S. N. Foley (Ed.): DBSec 2019, LNCS 11559, pp. 281–300, 2019.
https://doi.org/10.1007/978-3-030-22479-0_15

environments to explore the real-world asset digitally. The DT thus allows management of such an asset ranging from simple monitoring to autonomy. An essential part of the concept is the inclusion of the whole asset lifecycle. To integrate all lifecycle participants, the DT should provide comprehensive networking for its data, allowing it to be shared and exchanged [4].

Although the DT concept certainly advances digitization, it nevertheless poses new challenges in terms of IT security, especially in industrial ecosystems [10,18]. Most notably, security must be maintained during the exchange of DT data between different, non-trusting parties. For instance, consider the DT of a power plant. Synchronizing tasks between twins should uphold integrity to avoid manipulated operations on the power plant. Also, involved parties should not be able to read every shared data element (e.g. the manufacturer of the power plant need not know the plant's current status), resulting in confidentiality requirements. To the best of our knowledge, current DT frameworks do not permit secure data sharing. Bridging this gap, our work provides a framework introducing security-by-design in DT data sharing.

To achieve this goal, we consider Distributed Ledger Technology (DLT). DLT is the umbrella term for distributed transaction-based systems, shared among several independent parties in a network. Distributed Ledgers have built-in mechanisms for access control and asset management, including authentication and authorization mechanisms. We focus on permissioned distributed ledgers, which target enterprise usage by restricting access to fixed set of independent and semi-trusted participants. One of the main reasons for using a Distributed Ledger is disintermediation, replacing the need for trust in a third party or central operator through a replicated and integrity-preserving database. Inherent transparency and auditability are additional advantages over centralized solutions. Due to these properties, DLT is uniquely suited to solve the challenges of DT secure data sharing.

Accordingly, this work proposes a framework for secure DT data sharing across an asset's lifecycle and collaborating parties based on DLT. We contribute to the body of knowledge by offering a solution without a trusted third party (TTP) based on security-by-design. The remainder of this paper is organized as follows: Sect. 2 introduces the background of our work. Afterwards, we proceed to the description of the current problems in DT data sharing and name the resulting requirements for secure DT data sharing (Sect. 3). In Sect. 4, we provide a framework for secure DT data sharing for multiple parties based on DLT. To show practical relevance and the functionality of our framework, a use case is provided in Sect. 5. In Sect. 6, we evaluate our approach in terms of fulfillment of the stated requirements. To conclude, Sect. 7 sums up the main contributions and gives an outlook for future work.

2 Background

At present, the *Digital Twin* phenomenon is still in its infancy. Nevertheless, implementation and design of this concept are addressed to date, especially in the area of Industry 4.0. With strong focus on the industrial domain, the major part of research suggests DT implementation through AutomationML-formatted descriptive data of the real-world counterpart, e.g. [2,6,20]. The XML-based AutomationML (AML) format describes industrial assets and provides object-orientation for modeling the asset's physical and logical components [20]. Eckhart and Ekelhart [6] propose a framework for using a DT's simulation mode for security purposes such as pen testing. While these works focus on an initial development of a DT, the consideration of data sharing functions are still missing. However, exchanging data is vital for enabling the lifecycle integration and collaboration [4]. Our work builds on existing DT propositions, resulting in a concept that can be applied in a complementary way to enable secure DT data sharing.

Regarding DT data sharing, both the communication between lifecycle parties and the bidirectional communication between the DT and its real-world asset counterpart need to be considered. Bidirectional communication consists of the DT's instructions for the asset and the asset's status update for the DT. To uphold integrity among multi-domain DT models, Talkhestani et al. [21] offer a solution. They detect model changes by applying anchor points, and upon detection synchronize the DT while keeping model dependencies consistent. However, this includes drawbacks such as the manual creation of anchor points and reliance on a Product Lifecycle Management (PLM) system, while our solution offers platform-independence. Security aspects, such as the guarantee for all lifecycle partners to access the data while upholding confidentiality, are not considered to date, but integrated in our solution.

DT management is a form of enterprise asset management, which is one of the prime use cases of Distributed Ledgers [1]. Distributed Ledgers are able to track events and provenance information along an asset's lifecycle and increase transparency for all participants. For example, Litke et al. [12] studied the benefits of Distributed Ledgers for different actors in supply chain asset management, a research area closely related to DT asset management. In another study, Meroni and Plebani [14] investigate how the blockchain technology can be used for process coordination among smart objects. Smart objects are similar to DTs in that they are applied for monitoring physical artifacts. An issue with their proposed approach is that sensor data is also stored on the blockchain, which can be detrimental to performance and scalability. We consider this issue and provide a solution to overcome this obstacle.

3 Problem Statement

On the one hand, DTs should facilitate the access to asset information for different stakeholders along its lifecycle [17]. It is a task which enables feedback loops, while stepping towards a circular economy [3]. On the other hand, the involved parties do not necessarily trust each other, resulting in a confidentiality dilemma. A useful example is given in [13]: Two separate standalone DTs exist for a single device instance, one for the manufacturer and the other at the customer site – due to information security reasons. Additionally, current works state that enterprise infrastructures need to overcome the following obstacles to provide secure DT data sharing:

– application of different tools [13,24]
– usage of various data formats [13]
– missing standards [4]
– broken information flow across lifecycle phases [13,24]
– clarification of the ownership of information [13].

This calls for a holistic approach that provides confidentiality and integrity, two central security dimensions in networks [26].

3.1 Digital Twin Model

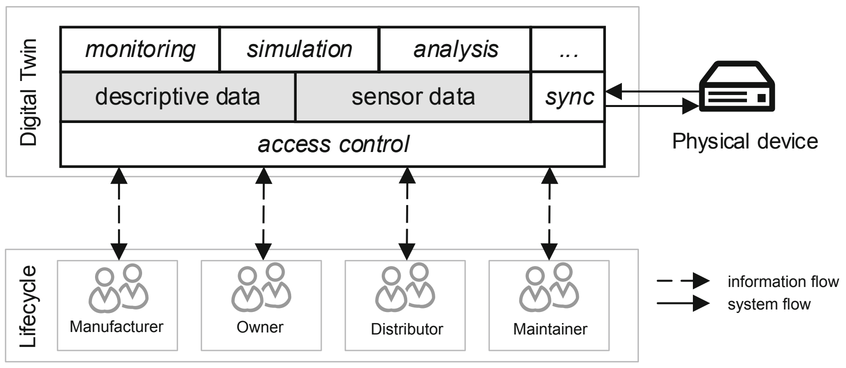


Fig. 1. Overview of the asset lifecycle participants interacting with the DT.

Figure 1 illustrates DT data sharing and an exemplary set of lifecycle stakeholders. The depicted DT model comprises different *capabilities* and two types of asset-specific data. *Descriptive data* refers to static properties of the device and infrequently changing state information. This data is mainly produced by users. *Sensor data* occurs frequently and should be available in near real-time. It is generated by sensors of the physical asset or in its proximity, which provide valuable information on the asset's environmental conditions. Moreover, data of both types needs to be synchronized with the physical counterpart. Therefore, the *sync* capability compares the state of the DT to its real-world counterpart and resolves possible discrepancies.

The *access control* capability provides authentication and authorization modules to enable data sharing of involved parties without hampering confidentiality. The *monitoring*, *simulation* and *analysis* capabilities represent advanced operations of the DT. Depending on the extent of the operations present in a DT, DT status data can be returned to the participant or the real-world counterpart's state can be modified.

The depicted information flows show how information about the physical device is gathered from and sent to the lifecycle parties. Generally, the type of data accessed and shared by the different lifecycle parties depends on the real-world twin, the parties' roles in its lifecycle and thus, the specified access control mechanisms in the DT. The system flows represent necessary bidirectional synchronization between the DT and its real-world counterpart as stated in Sect. 2. Both flows contribute to making the data sharing activities of the involved parties traceable. This enables feedback from the latest stages of the asset lifecycle to the earliest ones [17].

3.2 A Formal Basis for Secure Digital Twin Data Sharing

Although a methodological literature analysis to establish requirements is the state-of-the-art approach, it is currently not sensible to carry out with regard to our research focus. On the one hand, this is due to the fact that only a small number of publications exist. In addition, data sharing has not yet been a focus in DT literature to date. Moreover, security-by-design concepts have not been considered yet. Therefore, we establish a formally valid basis in order to create a uniform understanding of DT data sharing. To derive the requirements, the

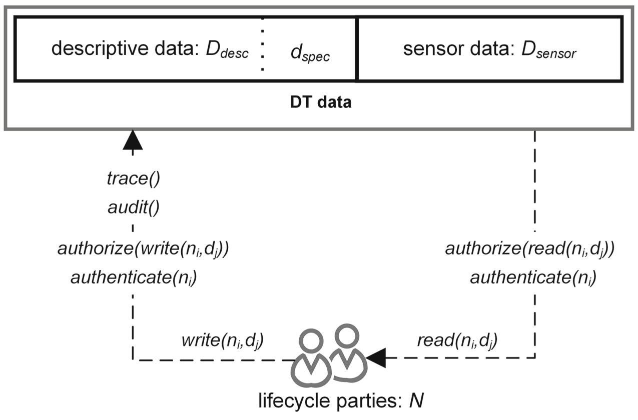


Fig. 2. Control flows for a single DT.

mechanisms to achieve the central goal of **secure DT data sharing** have to be examined in detail. Figure 2 illustrates the formal functions required to achieve this goal, which are also described hereafter.

DT Data: We see DT data twofold: At first, there is a set of descriptive data elements $D_{desc} := \{d_1, ..., d_m\}$ varying from documents to models or analytic outcomes. Its essential data element is the specification of the DT $d_{spec} \in D_{desc}$. The second set contains environmental, device-produced data, namely sensor data $D_{sensor} := \{d_1, ..., d_n\}$, whereby $D_{desc} \setminus D_{sensor}$.

Sharing: A finite set of lifecycle parties $N := \{n_1, ..., n_k | k \geq 2\}$ can share the respective data elements of D_{desc} (write operation) or access the data elements of D_{desc}, D_{sensor} (read operation). This results in the following necessary functions: $write(n_i, d_j | d_j \in D_{desc})$ and $read(n_i, d_j | d_j \in D_{desc} \vee d_j \in D_{sensor})$.

Note that $1 \leq i \leq k$ as well as $j \begin{cases} 1 \leq j \leq m & \text{if } j \in D_{desc} \\ 1 \leq j \leq n & \text{if } j \in D_{sensor} \end{cases}$.

Security-by-design: Security-by-design infers introducing security mechanisms at the very beginning of a system's design [22]. In terms of DT data and sharing security, data integrity and confidentiality mechanisms are of special interest. Confidentiality in terms of securing data from view of non-trusted third parties can be reached by access control mechanisms [19]:

authentication:

$$authenticate(n_i)$$

authorization:

$$authorize(read(n_i, d_j))$$

$$authorize(write(n_i, d_j))$$

Integrity of data can be achieved by auditability and traceability of write operations. Given D_{desc} as the origin set of data, D'_{desc} is the set of data after a data element d_j is added to the origin set. The following functions can cover integrity aspects:

auditability:

$$audit() : D_{desc} \rightarrow D'_{desc} \iff write(n_i, d_j) \quad \wedge$$

$$D_{desc} \nrightarrow D'_{desc} \iff \neg write(n_i, d_j)$$

traceability:

$$trace() : D_{desc} \rightarrow D'_{desc} \implies D_{desc} \circ D'_{desc}$$

Thereby, auditability guarantees that D_{desc} is transformed to D'_{desc} in case of an authorized write operation whereas other operations are not able to transform the data in any way. Traceability ensures that authorized writes of data elements and thus, transformations of D_{desc} to D'_{desc}, are chained up. In conclusion, data integrity is ensured as the data cannot be manipulated or tampered with in retrospect.

3.3 Requirements for Secure DT Data Sharing

To provide a sound solution for secure DT data sharing, the following requirements were derived from the formal basis and the aforementioned challenges identified in the literature analysis.

R1. Multi-party Sharing. To enable lifecycle inclusion, a vital characteristic of the DT paradigm [4], the multiple stakeholders N involved in the lifecycle have to be considered. As described in Fig. 1, parties can vary from manufacturer to maintainer. However, all involved parties are pre-registered and therefore determinable.

R2. Data Variety Support. At the heart of the DT lie the relevant digital artifacts D_{desc}, D_{sensor}, which vary from design and engineering data to operational data to behavioral descriptions [4]. Thus, different data types and data formats [13] need to be supported during data sharing. For instance, Schroeder et al. claim that using the semi-structured AutomationML format to model attributes related to the DT (d_{spec}) is very useful for DT data exchange [20]. In addition to semi-structured data, structured data (e.g. sensor tuples in D_{sensor}, database entries) and unstructured data such as human-readable documents can be asset-relevant and shared via the DT.

R3. Data Velocity Support. Often, DT data is distinguished between descriptive, rather static data, and behavioral, more dynamic data (see Fig. 1). The latter changes with time along the lifecycle of the real-world counterpart [20]: With each lifecycle stage the asset-related information evolves, resulting in different versions and a dynamic information structure [17]. Naturally, dynamic data includes sensor data D_{sensor} – which mostly refers to the actual state of the real-world counterpart [8]. While the infrequently changing data D_{desc} might not require high throughput, sensor and dynamic data D_{sensor} accrues in intervals ranging from minutes to milliseconds. Therefore, the solution must support high throughput and low sharing latency for efficient sharing of dynamic data – thus supporting data velocity.

R4. Data Integrity and Confidentiality Mechanisms. An important requirement is taking into account data security features, especially integrity and confidentiality. At first, this requirement aims at safeguarding data integrity to avoid wrong analytic decisions based on manipulated data. It can be ensured by $audit()$ and $trace()$ mechanisms. The second main security objective is to avoid confidentiality and trust problems while enabling multi-party participation. This calls for restricted data access dependent on the party through $authenticate()$ and $authorize()$ functions, while ideally keeping the effort for user registration low. Different levels of confidentiality should be possible for different data elements. For instance, D_{sensor} might need a lower level of protection than D_{desc}, as the latter might include sensitive corporate information such as blueprints. Detailed $authorize()$ functions, providing access-restrictions for each data element, can cover this aspect.

R5. Read and Write Operations. To interact with DT data, a DT data sharing solution must provide $read()$ and $write()$ data operations for the sharing parties. The allowance of operation modes for the data elements should be chosen carefully for each party to ensure **R4** (cf. Fig. 2).

Overall, we do not claim that these requirements are complete. There may be other requirements of importance, but regard these as essential for the following reasons. On the one hand, these requirements were found to be mentioned most often in the reviewed literature, while others were less frequently mentioned and are therefore considered of lower importance (see Sect. 6.2 for further explanation). On the other hand, the stated requirements were also the main focus in various practitioners reports (e.g. [9,16,25]) and during discussions with experts.

4 Solution Architecture

In order to develop a framework for secure DT data sharing, we first evaluate the suitability of DLT in Sect. 4.1. Afterwards, Sect. 4.2 explains the system architecture and Sect. 4.3 explains how the various data types are stored. Section 4.4 details the inclusion of the DT capabilities as part of the DLT solution. Finally, Sect. 4.5 explains the initial setup procedure for our framework.

4.1 Technology Selection

To develop a solution architecture, we first evaluate different data storage solutions' properties to select the technology best suited to fulfill the requirements.

A centralized solution could be created in the form of a portal, operated by a third party or the operator of the twinned device. This requires trust of the participating parties towards the portal maintainer, as the maintainer could manipulate data or revoke access to the DT for other parties. A distributed approach jointly operated by all participants could solve this trust issue. Distributed Ledgers represent such a distributed solution. They permit verifiable decentralized execution of business logic via *smart contracts*, ensuring that rules and processes agreed upon by the lifecycle participants are followed.

We evaluate the applicability of Distributed Ledgers to our DT data sharing requirements based on the blockchain applicability evaluation framework by Wüst and Gervais [28]. As illustrated in Fig. 1, there is a need to store various types of data as part of the DT state. Multiple parties interact with the twin during its lifecycle who do not fully trust each other. These writers are usually known in advance or change infrequently (i.e. the maintenance service provider changes). These characteristics lead to the choice of a public or private permissioned blockchain in the framework [28]. In our case, this choice depends on whether public auditability is required or not. While use-case dependent, we focus on private permissioned blockchains for the rest of this paper. If needed, public read-only access to blockchain data can be enabled during implementation for most permissioned blockchain frameworks (i.e. through a REST API).

4.2 System Architecture

The proposed DLT-based architecture for secure DT data sharing is shown in Fig. 3. Every participant runs three components: a node of a **Distributed Hash Table (DHT)**, a node of the **Distributed Ledger** and a **client application**. The DHT and Distributed Ledger make up the shared data storage, while the client application is responsible for the user interface and backend logic for retrieving and processing the data stored on the ledger and DHT. For owners

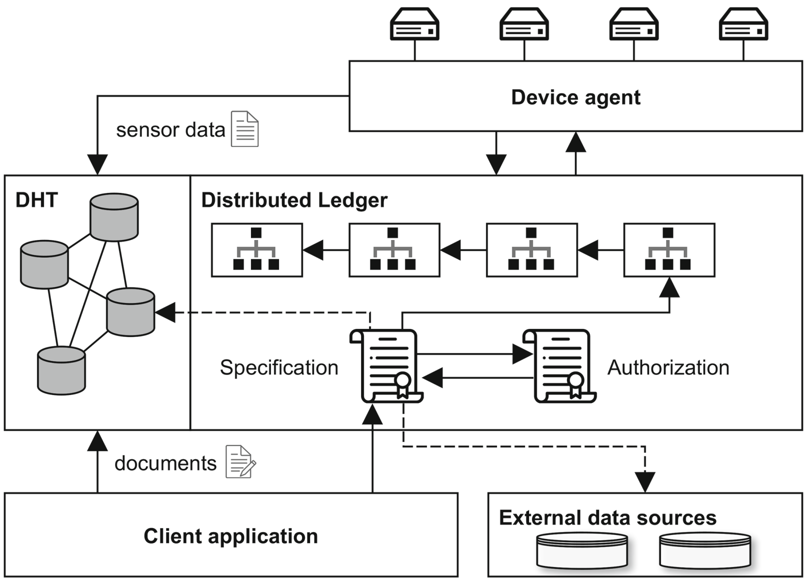


Fig. 3. DLT-based architecture for DT data sharing.

of twinned physical devices, a **Device agent** manages the physical devices and coordinates their interactions with the system. As part of operational technology, the Device agent functions as a bridge between the cross-organizational asset management system and the physical devices controlled by a single organization.

Data storage systems based on distributed ledgers have two ways of storing data: on-chain and off-chain [29]. On-chain storage is restricted to transactions and the internal state storage of smart contracts. Due to full replication of on-chain data, items larger than a few kilobytes in size need to be stored in a different, off-chain location. Using a traditional database would however result in a single point of failure or reintroduce a trusted party.

For this reason, we resort to a structured DHT for large data items. DHTs are distributed key-value stores, where all key-value pairs are mapped to one or more nodes. The DHT entries can be linked to the corresponding on-chain asset based on the DHT key hash. By storing the hash on the blockchain, integrity of the off-chain data can be verified after retrieving it from the DHT. To maintain confidentiality and availability, data stored on the DHT is encrypted, sharded and replicated. Correspondingly, an access control mechanism is needed to allow authorized parties to access the data. The k-rAC scheme illustrates how a DHT can implement the required functionality [11]. In k-rAC, access control is implemented using access control lists (ACL) stored along with each key-value pair on the DHT. We propose reusing the Distributed Ledger's public key identities for DHT authentication. A symmetric key is used for encryption, which is then available to authorized parties by encrypting it with their public key. The encrypted access keys are distributed with each data item's ACL. Manipulation of the ACL is prevented by requiring a quorum of $2k + 1$ nodes for write operations, where k is the number of tolerated malicious nodes.

4.3 Data Storage

There are two types of **descriptive data** that need to be stored by the system: a machine-readable specification and device-related unstructured data (i.e. human-readable documents). The **specification** includes a description of the device's hardware components as well as their functions. The DT's physical properties are derived from this specification. For our work we assume that AML is used to describe the physical asset. The AML specification is stored on the ledger in a modifiable way. This approach guarantees that updates to the device specification are observed by all parties. Distributed Ledgers can store complex modifiable state by using *smart contracts*. We thus refer to the resulting contract as the *specification contract*.

Unstructured data can be uploaded to the system and may subsequently be annotated or modified by other parties. Due to its size it cannot easily be parsed and stored in contracts. For this reason, it is stored off-chain and registered in the smart-contract with a document title and a hash of the contents. To update a document, a new version must be uploaded to the DHT and the smart contract reference updated. This ensures that changes to the documents are traceable.

Sensor data needs to be stored off-chain due to its frequent updates and the considerable amount of generated data. A history of the sensor data is kept to allow for further analysis, e.g. predictive maintenance or troubleshooting. The link to the on-chain data is established via a pointer to the off-chain storage location, stored on-chain in the specification contract. To avoid having to update the storage location hash every time new sensor data is uploaded to the DHT, we take advantage of *DHT feeds*. This concept is inspired by the Ethereum network's DHT Swarm [7]. In Swarm, a feed is defined by a feed manifest with a unique address on the network. The feed manifest's owner (i.e. the physical device) is the only user permitted to upload signed and timestamped data to this address. Any data format can be used and a history of uploaded data is kept. The DHT feed enables frequent sensor data sharing without having to update an on-chain reference. Based on the feed, the client application may compare sensor updates with expected values derived from the specification contract to detect anomalies. Additionally, there is no need for directly accessing the physical device, which may reside in a protected network. Instead, data updates are readily available on the DHT for authorized participants.

Many organizations also have additional internal data sources or microservices that provide structured data relevant to the Digital Twin. These data sources can be included in the twin by adding references (i.e. an URI) to the DT specification contract. This allows inclusion of legacy data sources and complex data which cannot easily be stored on a DHT (i.e. relational data). If the external data source requires authentication, it is the responsibility of the data source provider to ensure access rights for the DT ledger's identities.

Listing 1.1 shows a pseudocode representation of the data types stored in the specification contract. The syntax is inspired by Ethereum's Solidity smart contract programming language. All data stored on the contract is readable by all lifecycle participants. Besides general device metadata, the contract also includes a program call queue for interaction with the physical device's program interfaces (see also Sect. 4.4). Since smart contracts must be deterministic and thus cannot interact with files, the AML specification is stored in a string variable. This variable can later be parsed and modified, as illustrated in Sect. 4.4. Hash references to new original documents on the DHT are kept track of in the `documents` mapping. The hash serves as an identifier, while the `document` struct provides metadata. Updated versions of each document are stored in the `documentVersions` mapping. The componentID and corresponding feed reference of the sensor data stream on the DHT are stored in the `sensorFeeds` mapping.

```
/* metadata and specification*/
string deviceName
string deviceID
string deviceAML
string[] callProgramQueue

/* additional descriptive data */
struct Document {
    uint timestamp
    string description
    address owner
}

struct ExternalSource{
    string URI
    address owner
}

mapping(string=>Document) documents
mapping(string=>string[]) documentVersions
ExternalSource[] externalSources

/* sensor data */
mapping(string=>string) sensorFeeds
```

Listing 1.1. Data structures of the specification contract

```
/* descriptive data interfaces */
function addDocument(document)
function addDocumentVersion(string hash)
function removeDocument(string hash)

function addExternalSource(string URI)
function removeExternalSource(string URI)

/* sensor data interfaces */
function addSensorFeed(string componentID,
      string reference)

function removeSensorFeed(string componentID)

/* interaction with the specification */
function insertAML(string amlCode, string
      parentID, string afterID)
function removeAML(string ID)
function callProgram(string programName,
      string parameters[])
```

Listing 1.2. Function interfaces of the specification contract

4.4 Capabilities

We focus on the three capabilities required for accessing and publishing DT data: *DT interaction*, *access control* and *sync*.

DT interaction refers to the information flows in Fig. 1, which allow users to interact with the twin's data. The specification contract implements this functionality. It allows users to read and potentially modify the DT instance. The relevant interfaces that can be called with transactions are shown in Listing 1.2. New or updated references to documents may be appended by any authorized user. The same applies to external data sources and sensor feed references to the DHT. The specification can be manipulated by inserting or removing specific AML segments, which are identified by their ID. To determine the position of a new AML code segment in the AML document, the parent ID and the ID of the preceding element need to be passed as parameters. The twin's program interfaces for setting device parameters can be accessed via `callProgram`. This function checks authorization, finds the requested program in the AML specification and places it in a queue for the Device agent to retrieve. The agent then forwards the program call to the device for execution.

The *access control* capability is responsible for authentication and authorization of user interactions with the DT data. For user authentication, accounts are created on the blockchain and represented by their public key. An initial solution could be provided by the framework's built-in identity management, for example Hyperledger Fabric's Membership Service Provider (MSP) [1]. The MSP lists the certificate authorities who may issue digital identities for the Distributed Ledger. The same identity can then be reused for authentication in the DHT. Authorization is realized in a separate access control smart contract. Any protected interaction with the Digital Twin is first authorized through that contract. Such interactions are for example modifications of the twin's properties, like changing parameters or modifying its specification. A query from the client application provides an identity to the specification contract, which then interacts with the authorization contract to determine if the user is allowed to perform the action. Authorization is then granted or denied based upon a stored role-permission mapping. Accordingly, the contract's interfaces are based upon a Role-based Access Control (RBAC) scheme. We do not describe the access control contract in detail here, as there are other works describing blockchain-based access control schemes [5].

The *sync* capability requires regular interaction between the Device agent and the Distributed Ledger. For synchronization, the Device agent pulls updates from the real-world asset and uploads them to the off-chain DHT sensor data feed. The Device agent monitors the ledger and pushes any modifications instructed by committed on-chain transactions to the asset. The synchronization interval depends on the use case.

Other DT capabilities like *monitoring, simulation* and *analysis* can be executed off-chain by interacting with the local copy of the ledger. Simulation or analysis instructions and results can be shared on the ledger as documents. This would allow other parties to verify the results, should they desire to do so.

4.5 Setup Process

Initially, each lifecycle participant sets up one network node running both a DHT and a Distributed Ledger node. These serve as local replicas of ledger data and access points for off-chain data. They may also be used for transaction-based interaction with the smart contracts. Additionally, an identity provider must be set up to allow federated identities from all participating organizations based on public key certificates.

Once the network is set up, a Digital Twin instance can be created on the ledger by the device owner. The manufacturer should first provide the AML file to the owner, who then proceeds to set up a Digital Twin sharing instance on the ledger. The client application provides the interface to upload the file and create a smart contract based on it. Before uploading, the owner also needs to specify the access rights associated with the various parts of the specification. Although use case dependent, sensible default values could be *write* access by owner and maintainer and *read* access by everyone else.

In this way, any number of Digital Twin instances can be created by the various parties on the network. Each instance is represented by a specification contract. Subsequent modifications take place via authorized on-chain transactions and are stored as part of the contract's internal state. As a result, auditing the twin is possible by (actively or retroactively) monitoring smart contract transactions for anomalies.

5 Use Case

This chapter intends to show how the theoretical framework developed in Sect. 4 is traversed in a use case. To begin with, the overall setting of the use case is described in Sect. 5.1, while the subsequent Sect. 5.2 iterates the use case through the solution architecture. At last, a summary is given, focusing on the automation degree in data sharing and the reading operation (Sect. 5.3).

5.1 Setting

The setting is chosen close to reality. The asset, the real-world counterpart to the DT, is a bottling plant, where bottles are filled with beverages. The parties involved in the asset lifecycle are a manufacturer, an owner, a maintainer of the bottling plant and an external auditor that audits the safety of our bottling plant. For our use case, we consider the following scenario: The bottles are flooding due to a broken sensor in the bottling plant. Consequently, the maintainer detects the damage and changes the broken sensor in the bottling plant.

This entails the following shared data interactions. At first, the specification of the plant needs to be updated by replacing the broken sensor's specification entry with the newly added sensor. Additionally, the new sensor's data stream has to be integrated in place of the old sensor stream. Other documents concerning the maintenance task might also be shared, such as a maintenance report.

While the maintainer is the only party sharing data in this scenario, the owner should also be updated on the state of the bottling plant. Furthermore, the manufacturer needs to be informed that the sensor is broken, so that an analysis of the time and circumstances can be conducted. This way relevant insights for future plant manufacturing can be gained. Additionally, the external auditor needs to access the information about the maintenance task to review the procedure in terms of safety compliance.

5.2 Framework Iteration

This use case triggers a specific logical order of events in the framework, which are highlighted in Fig. 4 and described hereafter. The framework first comes into play when the maintainer replaces the broken sensor.

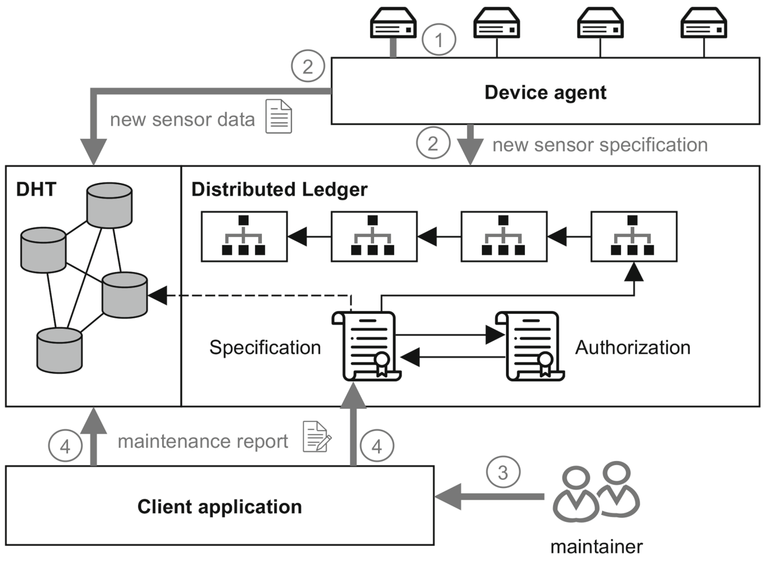


Fig. 4. Use case tailored architecture for DT data sharing.

1. All devices are connected with the **Device agent**, which registers the exchange of the broken sensor. Additionally, it gathers information about the new sensor.
2. Following the new sensor connection, the **Device agent** forwards the new incoming data stream of the sensor into the **DHT**. The location of the stored sensor stream in the **DHT** is registered by the **Device agent**.

   The **Device agent** then sends a transaction containing the new sensor specification to the **Distributed Ledger**. This transaction invokes the specification contract, resulting in several updates. First, the old sensor entry is removed and the new sensor specification given by the **Device agent** is added. Secondly, the storage location of the sensor stream on the **DHT** is added by a reference to the location. These three transactions concerning the specification are stored on the **Distributed Ledger**.
3. Having performed the maintenance task, the maintainer writes a maintenance report and pushes it onto the **Client application**.
4. The **Client application** adds the maintenance report by performing two actions. Firstly, it adds the report to the off-chain **DHT**. Secondly, it stores the reference to the **DHT** location of the report on the specification contract. Thereby, the location is added to the entry of the sensor specification.

5.3 Results

In a nutshell, the recognition of new sensor and the AML update with the new component is already accomplished by the **Device agent** without requiring human interaction. The new data stream is automatically forwarded to the **DHT** and the reference to the new storage location of the component's data stream is added to the specification contract. Additional unstructured non-specification data (e.g. the maintenance report) can be added manually. The **Client application** takes care of the necessary background work by inserting the file into the **DHT** and adding the respective storage reference into the specification contract.

All participating parties can view the latest transactions on the ledger – presented in a comprehensive way in the **Client application**. Advanced **Client applications** could also notify the user whenever an ledger update takes place.

Considering security, the advantages of this framework shine when compared to the alternative solution: A TTP could deliberately transfer shared information and know-how to rival enterprises. For instance, confidential sensor data or blueprints could be leaked to competitors, which may then deduce quality issues of the rival product. The service of the TTP could also be compromised by attackers, resulting e.g. in a violation of integrity so that the sharing parties receive inconsistent asset versions.

6 Evaluation

To evaluate our framework, Sect. 6.1 discusses the suitability of the framework in reference to the requirements. Finally, the results are discussed in Sect. 6.2.

6.1 Requirements Fulfillment

To sum up, our approach fulfills the requirements **R1**–**R5**. The following paragraphs explain how each requirement was addressed in our solution architecture.

R1. Multi-party Sharing. The main argument for using Distributed Ledgers is the involvement of multiple parties N who produce and consume data. Next to the ledger, our approach provides a client application for all parties that accesses the data on the ledger and the DHT. Therefore, our approach clearly fulfills **R1**.

R2. Data Variety Support. To enable the sharing of different data in various formats, our approach provides a central documentation and two storage options. The standardized asset description d_{spec} is included in the Distributed Ledger and serves as the basis of the DT within the specification contract. All other data of D_{desc} as well as the sensor data D_{sensor} are stored off-chain in the DHT. Moreover, each stored data element in DHT is registered in the central specification contract as a reference to the storage location of the data element. For instance, a sensor in the specification contract contains a reference to the storage position of its data stream in the DHT. Hence, **R2** is met.

R3. Data Velocity Support. Modern sensor data streams' frequency and volume exceed the performance characteristics of current Distributed Ledger frameworks. Since the data streams D_{sensor} do not describe main features of the DT (d_{spec}), they are stored off-chain in the DHT. This way, high throughput of D_{sensor} is supported, while the sharing latency is also kept low (seconds). The Distributed Ledger maintains verifiability by storing the hash reference to the data stream on the DHT in the specification contract. This ensures no loss in performance and data access through the DHT, supporting **R3**.

R4. Data Integrity and Confidentiality Mechanisms. With respect to data integrity, the Distributed Ledger attaches every new data element (*trace*()) and prevents manipulation of the data by replicating it among all involved parties. A manipulation would result in a version mismatch or loss of consensus and could be detected easily (*audit*()). The second storage component (DHT) also supports integrity by storing the respective hash values to the data. A manipulation of DHT data would also be detected by a mismatch between the hashes in the nodes (*audit*()). However, there remains the problem of adding non-valid data, which is a common issue in the area of DLT. Here, we rely on the parties' interest in sharing valid data and on mechanisms ensuring quality of input data that the respective responsible party applies.

In terms of data confidentiality, our approach ensures that the data is read only by authenticated and authorized parties. Authentication is ensured through lifecycle party login to the client application (*authenticate*()). Access control concerning the party and the data elements is realized through an ACL and encryption for off-chain data and an authorization smart contract for on-chain data (*authorize*()). In concrete terms, the ACLs specify access rights on a per-document basis, while the smart contract stores authorization information for all involved parties. Therefore, different confidentiality levels can be realized.

To conclude, our approach provides data integrity and confidentiality mechanisms (**R4**) – reinforcing data security in DT data sharing.

R5. Read and Write Operations. Read and write operations are managed through the Client application. For *read*() operations, the Client application fetches the requested data from the DHT and the ledger and presents the data in a comprehensive way adjusted for the demanding party. In case of a *write*() operation, the Client application triggers the right procedure to alter the smart contract with a transaction and uploads additional asset-relevant data beyond specification to the DHT. Consequently, our approach also fulfills **R5**.

6.2 Discussion

Keeping the requirements *variety* **(R2)** and *velocity* **(R3)** in mind, the question arises why data *volume* is not considered a requirement. As literature is currently not at consensus regarding the relevance of the Big Data feature *volume* [15] for Digital Twins, we consider explicit support for data volume to be non-necessary. Nevertheless, by storing documents off-chain, our approach can handle considerable amounts of data. Future implementations of our concept may conduct benchmark studies to explore scalability limits with regard to big data volumes.

It should be noted that our approach depends on multi-party participation. The more independent parties maintain the Distributed Ledger and DHT, the less vulnerable the data sharing is to manipulation. With regard to the access control capability, a decentralized identity management solution with a shared identity database could be an even more holistic, next-generation solution.

While we are aware that our approach currently lacks an implementation, we nevertheless believe that the use case shows suitability for practice. Future work will focus on implementing the framework. Here, challenges might include adjusting a DHT framework to support authorization and data feeds (although Swarm shows promise in this regard [7]), as well as selecting a suitable Distributed Ledger framework.

The Distributed Ledger and the concomitant smart contracts could also be handled in a different way. For instance, the AML could be transformed into classes and types in the smart contract, similar to the BPMN to Solidity transformation in [27]. However, the effort clearly outweighs the utility as AML is a very powerful standard allowing very complex descriptions. Moreover, not all of the hypothetically generated classes and functions might be needed. Plus, functions or classes might be newly added later on, which results in the need to re-create the smart contract as they are currently not represented in the smart contract. This clearly increases effort and downgrades utility.

Another issue is entailed by the possibility to directly alter variable values referring to an actual function in our current version of the ledger. For instance, consider a PLC device with various functions such as setting a conveyor belt's velocity (with an integer parameter). Without constraints, the changed velocity could exceed safety bounds. Safety threats like this one, be they malicious or accidental, need to be mitigated in a production system. Therefore, we suggest integrating safety and security rules as proposed in [6]. They could be integrated as part of the specification contract, with the Device agent checking conformance of program calls on synchronization.

With respect to the current problems hampering secure DT data sharing, our approach tackles the issues stated in Sect. 3 in the following ways:

- The usage of different tools that can be connected with our main data sharing approach (External data sources, Fig. 3) is possible (*application of different tools*)
- Our approach is tailored for the integration of data in multiple formats and variety as stated in Sect. 4.3 (*usage of various data formats*)
- An agreement only on the standard describing the asset (e.g. AML) is required to transform the main description of the asset into the specification smart contract, while other standardized or non-standardized data can still be shared via the DHT (*missing standards*)
- The proposed shared collaborative data basis is distributed among all involved parties and the information flow is universal across the lifecycle phases (*broken information flow across lifecycle phases*)
- The Distributed Ledger registers the data as well as the involved party sharing the data, while mechanisms such as access control (Authorization contract, Fig. 3) support confidentiality issues (*clarification of the ownership of information*).

To sum up, the major part of the identified issues in the literature referring to DT data sharing are diminished or solved by our approach.

7 Conclusion

DT data not only ties physical and virtual twin [23], it also enables integration of the whole asset lifecycle, which is essential for realizing the DT paradigm. Moreover, the exchange of asset-relevant data (DT data) is vital for achieving the effects of a feedback loop. Closing the feedback loop in turn favors the development of a circular economy.

However, maintaining data security becomes a major requirement when sharing DT data between multiple parties, especially as the parties do not necessarily trust each other. Our approach of applying DLT can clearly solve this issue and enable secure multi-party data sharing. It provides confidentiality through access control arranged by usage of a smart contract. Moreover, data integrity is implicitly supported through the immutability of the original data in the ledger.

To conclude, our approach fulfills the requirements **R1**–**R5** for secure DT data sharing. Nevertheless, there remain minor drawbacks that need to be addressed in future research (see Sect. 6.2). Our upcoming work will focus on implementing our theoretical concept to demonstrate its feasibility in practice.

References

1. Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, pp. 30:1–30:15. ACM, New York (2018). https://doi.org/10.1145/3190508.3190538
2. Banerjee, A., Dalal, R., Mittal, S., Joshi, K.P.: Generating digital twin models using knowledge graphs for industrial production lines. In: Workshop on Industrial Knowledge Graphs, No. June, pp. 1–5 (2017). http://ebiquity.umbc.edu/paper/html/id/779/Generating-Digital-Twin-models-using-Knowledge-Graphs-for-Industrial-Production-Lines
3. Baumgartner, R.J.: Nachhaltiges Produktmanagement durch die Kombination physischer und digitaler Produktlebenszyklen als Treiber für eine Kreislaufwirtschaft. In: Interdisziplinäre Perspektiven zur Zukunft der Wertschöpfung (2018). https://doi.org/10.1007/978-3-658-20265-1_26
4. Boschert, S., Heinrich, C., Rosen, R.: Next generation digital twin. In: Proceedings of TMCE 2018, No. May (2018). https://www.researchgate.net/publication/325119950
5. Di Francesco Maesa, D., Mori, P., Ricci, L.: Blockchain based access control. In: IEEE Blockchain Conference 2018, pp. 1379–1386 (2018). https://doi.org/10.1007/978-3-319-59665-5_15
6. Eckhart, M., Ekelhart, A.: Towards security-aware virtual environments for digital twins. In: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security - CPSS 2018, pp. 61–72 (2018). https://doi.org/10.1145/3198458.3198464
7. Ethereum Swarm Contributors: Swarm 0.3 documentation (2019). https://readthedocs.org/projects/swarm-guide/downloads/pdf/latest/
8. Glaessgen, E., Stargel, D.: The digital twin paradigm for future NASA and U.S. air force vehicles. In: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference (2012). https://doi.org/10.2514/6.2012-1818

9. Greengard, S.: Building a Better Iot (2017). https://cacm.acm.org/news/218924-building-a-better-iot/fulltext
10. ICS-CERT: Overview of cyber vulnerabilities. Technical report (2017). https://ics-cert.us-cert.gov/content/overview-cyber-vulnerabilities
11. Kieselmann, O., Wacker, A., Schiele, G.: k-rAC - a fine-grained k-resilient access control scheme for distributed hash tables. In: Proceedings of the 12th International Conference on Availability, Reliability and Security, ARES 2017, Reggio Calabria, Italy, pp. 1–43. ACM, New York (2017). https://doi.org/10.1145/3098954.3103154
12. Litke, A., Anagnostopoulos, D., Varvarigou, T.: Blockchains for supply chain management: architectural elements and challenges towards a global scale deployment. Logistics **3**(1) (2019). https://doi.org/10.3390/logistics3010005
13. Malakuti, S., Grüner, S.: Architectural aspects of digital twins in IIoT systems. In: Proceedings of the 12th European Conference on Software Architecture Companion Proceedings - ECSA 2018, pp. 1–2 (2018). https://doi.org/10.1145/3241403.3241417
14. Meroni, G., Plebani, P.: Combining artifact-driven monitoring with blockchain: analysis and solutions. In: Matulevičius, R., Dijkman, R. (eds.) CAiSE 2018. LNBIP, vol. 316, pp. 103–114. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92898-2_8
15. Negri, E., Fumagalli, L., Macchi, M.: A review of the roles of digital twin in CPS-based production systems. Procedia Manuf. **11**(June), 939–948 (2017). https://doi.org/10.1016/j.promfg.2017.07.198
16. Ovtcharova, J., Grethler, M.: Beyond the Digital Twin - Making Analytics come alive. visIT [Industrial IoT - Digital Twin], pp. 4–5 (2018). https://www.iosb.fraunhofer.de/servlet/is/81714/
17. Ríos, J., Hernández, J.C., Oliva, M., Mas, F.: Product avatar as digital counterpart of a physical individual product: literature review and implications in an aircraft. In: Advances in Transdisciplinary Engineering (2015). https://doi.org/10.3233/978-1-61499-544-9-657
18. Rubio, J.E., Roman, R., Lopez, J.: Analysis of cybersecurity threats in industry 4.0: the case of intrusion detection. In: D'Agostino, G., Scala, A. (eds.) CRITIS 2017. LNCS (LNAI and LNB), vol. 10707, pp. 119–130. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-99843-5_11
19. Sandhu, R.S., Samarati, P.: Access control: principles and practice. IEEE Commun. Mag. (1994). https://doi.org/10.1109/35.312842
20. Schroeder, G.N., Steinmetz, C., Pereira, C.E., Espindola, D.B.: Digital twin data modeling with automationML and a communication methodology for data exchange. IFAC-PapersOnLine **49**(30), 12–17 (2016). https://doi.org/10.1016/j.ifacol.2016.11.115
21. Talkhestani, B.A., Jazdi, N., Schloegl, W., Weyrich, M.: Consistency check to synchronize the Digital Twin of manufacturing automation based on anchor points. Procedia CIRP (2018). https://doi.org/10.1016/j.procir.2018.03.166
22. Tankard, C.: The security issues of the Internet of Things. Comput. Fraud Secur. **2015**(9), 11–14 (2015). https://doi.org/10.1016/S1361-3723(15)30084-1
23. Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., Sui, F.: Digital twin-driven product design, manufacturing and service with big data. Int. J. Adv. Manuf. Technol. **94**(9–12), 3563–3576 (2018). https://doi.org/10.1007/s00170-017-0233-1
24. Uhlemann, T.H., Lehmann, C., Steinhilper, R.: The digital twin: realizing the cyber-physical production system for industry 4.0. Procedia CIRP (2017). https://doi.org/10.1016/j.procir.2016.11.152

25. Usländer, T.: Engineering of digital twins. Technical report, Fraunhofer IOSB (2018). https://www.iosb.fraunhofer.de/servlet/is/81767/
26. Voydock, V.L., Kent, S.T.: Security mechanisms in high-level network protocols. ACM Comput. Surv. (1983). https://doi.org/10.1145/356909.356913
27. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
28. Wüst, K., Gervais, A.: Do you need a blockchain? In: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), pp. 45–54 (2018). https://doi.org/10.1109/CVCBT.2018.00011
29. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. In: Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, pp. 182–191. IEEE (2016). https://doi.org/10.1109/WICSA.2016.21

1.2 Ethertwin: Blockchain-based secure digital twin information management [P2]

Status:	published
Publication:	Information Processing & Management
Submitted:	15 May 2020
Revised:	26 October 2020
Accepted:	27 October 2020
Citation:	Benedikt Putz, Marietheres Dietz, Philip Empl, and Günther Pernul. 2021. Ethertwin: Blockchain-based secure digital twin information management. Information Processing & Management 58, 1 (2021).

Journal Description: Information Processing and Management publishes cutting-edge original research at the intersection of computing and information science concerning theory, methods, or applications in a range of domains, including but not limited to advertising, business, health, information science, information technology marketing, and social computing.

Information Processing and Management 58 (2021) 102425

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).

https://doi.org/10.1016/j.ipm.2020.102425
Received 15 May 2020; Received in revised form 26 October 2020; Accepted 26 October 2020

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, Otoum, 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	• Sketches	• Sensor data	• Condition of the components
	• Blueprints	• System logs	
	• Manuals	• Maintenance reports	• Component's location
	• Design models	• Simulations	
Involved parties	Owner, Manufacturers, Distributors	Owner, *Manufacturers, Distributors*, Maintainers	Owner, *Manufacturers, Distributors*, 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

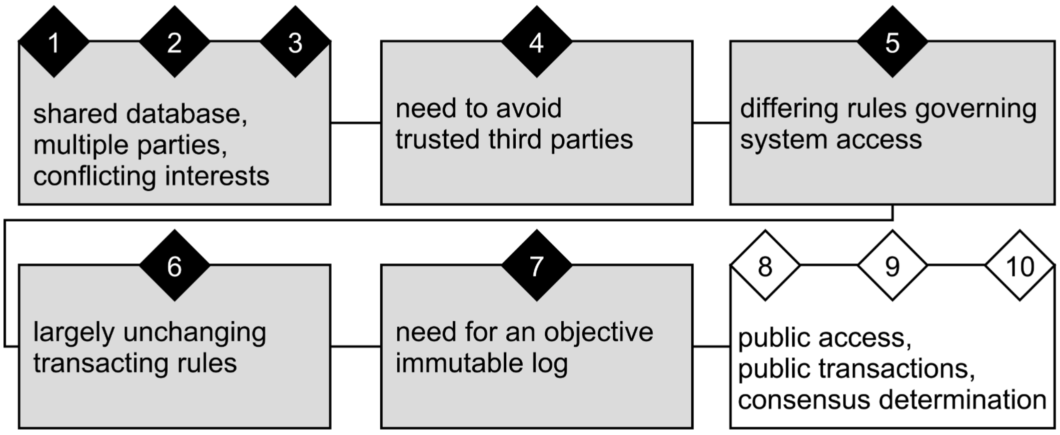

1
2
3
shared database, multiple parties, conflicting interests
4
need to avoid trusted third parties
5
differing rules governing system access
6
largely unchanging transacting rules
7
need for an objective immutable log
8
9
10
public access, public transactions, consensus determination

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. Related research on data auditing has shown that blockchain technology is able to remove the need for trusted 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

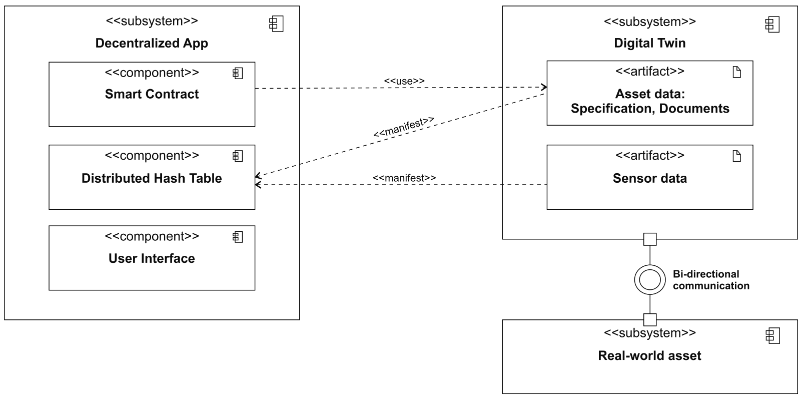


Fig. 2. Component Diagram describing the Digital Twin Sharing Context.

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 Swarm[1] are based on the established and secure DHT routing technology S/Kademlia Baumgart and Mies (2007) and integrate well with blockchains such as Ethereum[2].

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

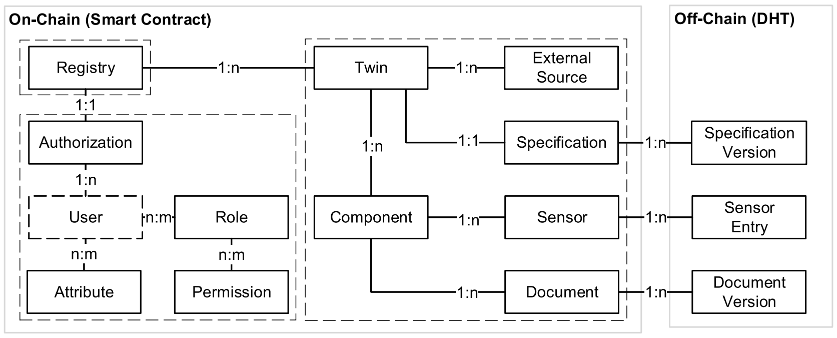


Fig. 3. Entity Relationship Model of the DApp.

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* $U := \{u_1, \cdots, u_n\}$. In our hybrid access control approach, each user can have one *Role* $R := \{r_1, \cdots, r_n\}$ as well as several *Attributes* $A := \{a_1, \cdots, a_n\}$ per DT. *Components* $C := \{c_1, \cdots, c_n\}$ serve a special purpose in this access control model, as they are used for modeling *Attributes*: $A := a_1, \cdots, a_n \mid a_i = c_1 \vee \ldots \vee c_n$.

To continue, *Permissions* $P := \{p_1, \cdots, p_n\}$ 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 $O := \{o_1, \cdots, o_n\}$ and the allowed operation $Op := \{op_1, \cdots, op_n\}$. Objects are always associated to a component to link the asset-relevant data to the component they belong: $o \rightarrow c \mid o \in O \wedge c \in C$. 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 $UA = U \times$

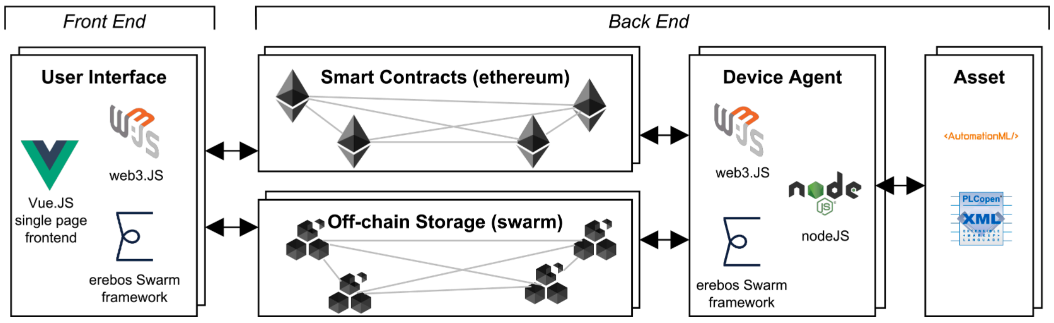


Fig. 4. Technologies used in our prototypical implementation of DT data sharing.

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

$$assign\_users(r,a) = u \in U \mid (u,r) \in UR \land (u,a) \in UA$$

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:

$$assign\_permissions(r,a) = p \in P \mid (p,r) \in PR \land p \rightarrow c \mid c \equiv a$$

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.JS[3] – 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 module[4] 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

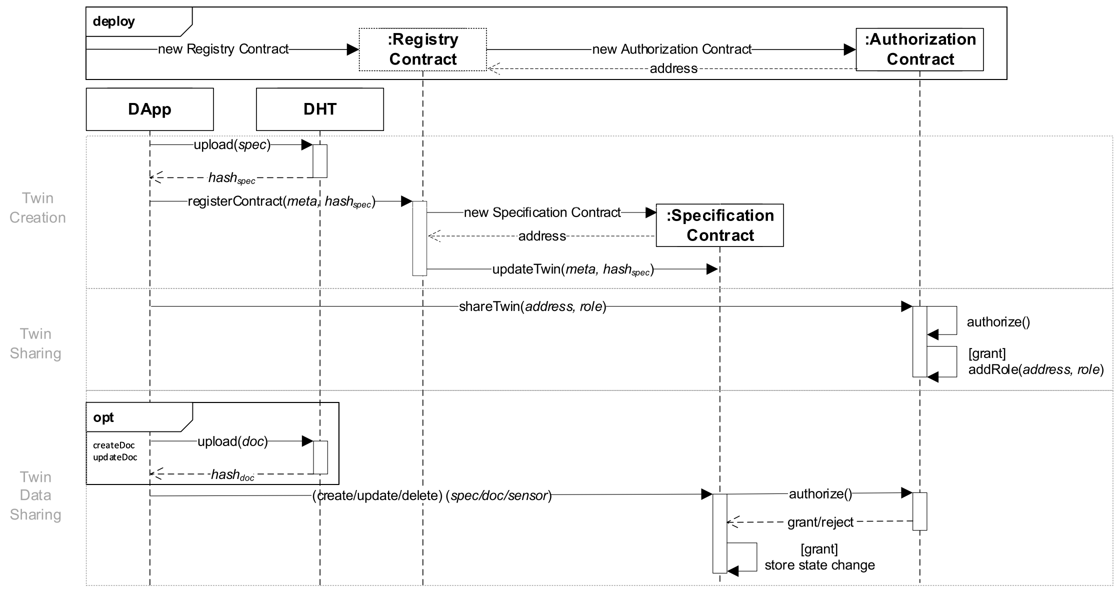


Fig. 5. Sequence diagram showing initial deployment and user interactions with the smart contracts and DHT.

The **Device Agent** bidirectionally synchronizes the DT's underlying **Asset** with the decentralized DT on Ethereum and Swarm. It runs as a node.JS[5] 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 **C**reate/**R**ead/**U**pdate/**D**elete permissions. ∼: Permission depends on presence of component attribute.

Permission	Twin				Document				Sensor			
	C	*R*	*U*	*D*	*C*	*R*	*U*	*D*	*C*	*R*	*U*	*D*
Device	×	✓	×	×	×	✓	×	×	×	✓	✓	×
Owner	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Manufacturer	×	✓	×	×	∼	∼	∼	×	×	∼	×	×
Maintainer	×	✓	×	×	∼	∼	∼	×	∼	∼	∼	×
Distributor	×	✓	×	×	∼	∼	∼	×	×	×	×	×

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.

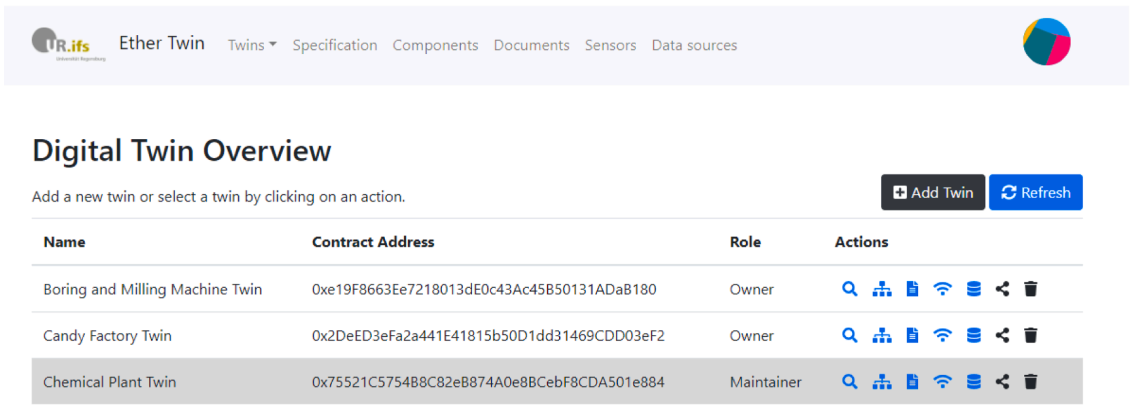
Encryption. All data is AES-256-encrypted before being uploaded to the Swarm DHT. Permissions are enforced by sharing a public-key encrypted version of the symmetric file key. Since Ethereum uses public keys based on elliptic curve cryptography, we rely on the Elliptic Curve Integrated Encryption Scheme (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.

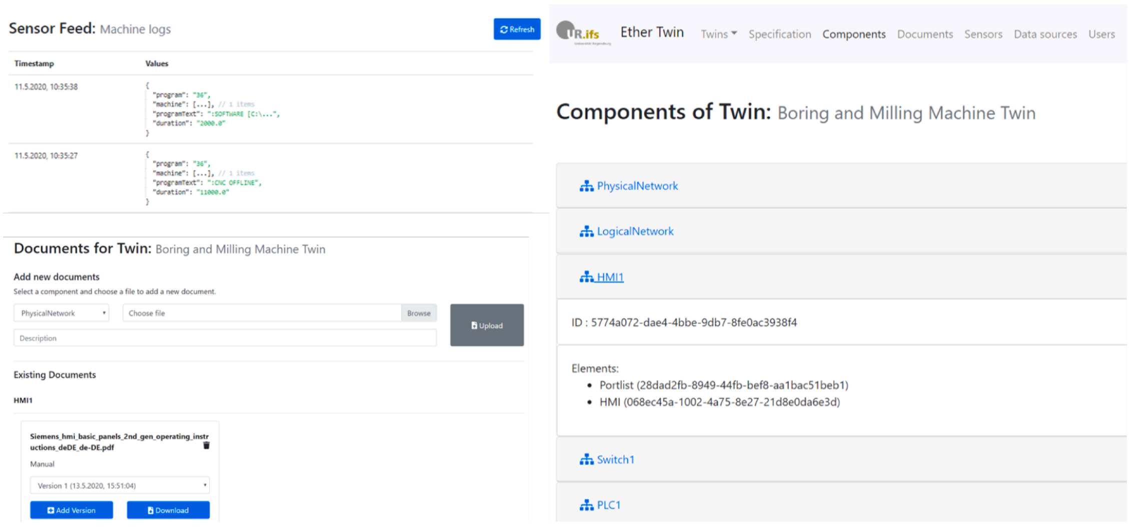


Digital Twin Overview

Add a new twin or select a twin by clicking on an action.

Name	Contract Address	Role	Actions
Boring and Milling Machine Twin	0xe19F8663Ee7218013dE0c43Ac45B50131ADaB180	Owner	
Candy Factory Twin	0x2DeED3eFa2a441E41815b50D1dd31469CDD03eF2	Owner	
Chemical Plant Twin	0x75521C5754B8C82eB874A0e8BCebF8CDA501e884	Maintainer	

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



Sensor Feed: Machine logs

Timestamp	Values
11.5.2020, 10:35:38	{ "program": "36", "machine": [...], // 1 items "programText": ":SOFTWARE [C:\...", "duration": "2000.0" }
11.5.2020, 10:35:27	{ "program": "36", "machine": [...], // 1 items "programText": ":CNC OFFLINE", "duration": "11000.0" }

Documents for Twin: Boring and Milling Machine Twin

Add new documents

Select a component and choose a file to add a new document.

Choose file

Description

Existing Documents

HMI1

Siemens_hmi_basic_panels_2nd_gen_operating_instructions_deDE_de-DE.pdf

Manual

Version 1 (13.5.2020, 15:51:04)

Components of Twin: Boring and Milling Machine Twin

PhysicalNetwork

LogicalNetwork

HMI1

ID : 5774a072-dae4-4bbe-9db7-8fe0ac3938f4

Elements:
- Portlist (28dad2fb-8949-44fb-bef8-aa1bac51beb1)
- HMI (068ec45a-1002-4a75-8e27-21d8e0da6e3d)

Switch1

PLC1

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.

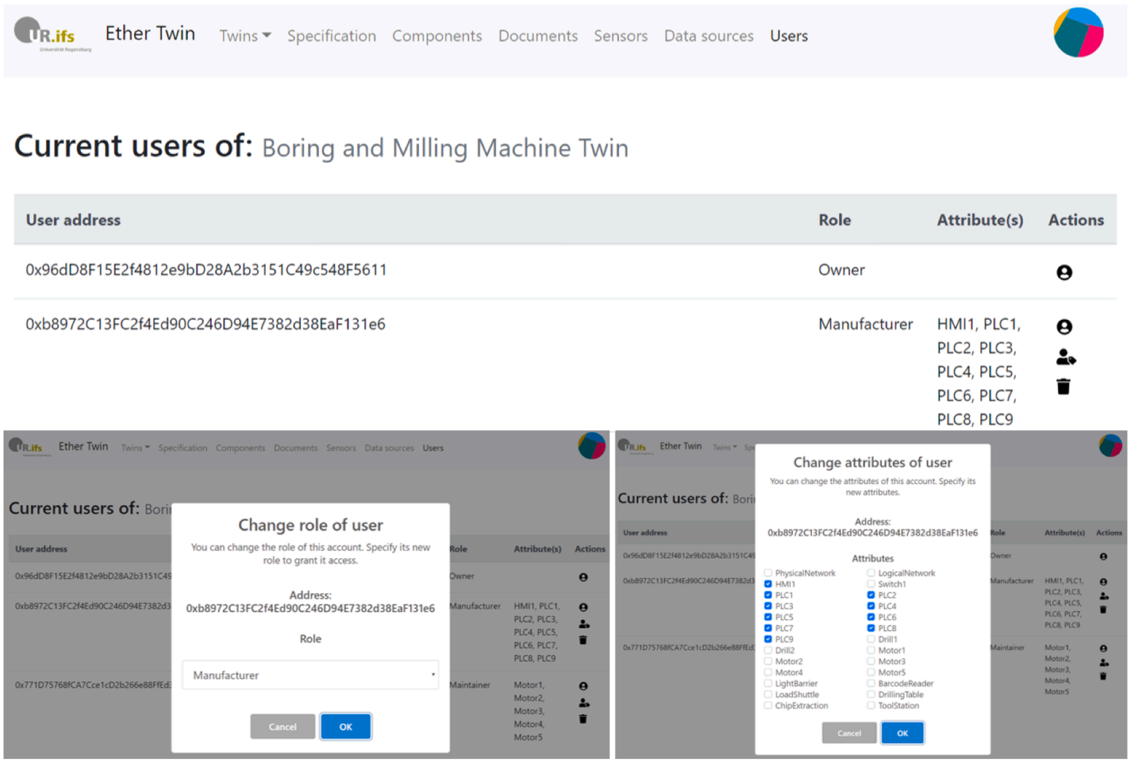
6.1. Prototype

The EtherTwin prototype is available on GitHub[6], 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



Current users of: Boring and Milling Machine Twin

User address	Role	Attribute(s)	Actions
0x96dD8F15E2f4812e9bD28A2b3151C49c548F5611	Owner		
0xb8972C13FC2f4Ed90C246D94E7382d38EaF131e6	Manufacturer	HMI1, PLC1, PLC2, PLC3, PLC4, PLC5, PLC6, PLC7, PLC8, PLC9	
0x771D75768fCA7Cce1cD2b266e88FfEd	Maintainer	Motor1, Motor2, Motor3, Motor4, Motor5	

Change role of user

You can change the role of this account. Specify its new role to grant it access.

Address:
0xb8972C13FC2f4Ed90C246D94E7382d38EaF131e6

Role: Manufacturer

Cancel OK

Change attributes of user

You can change the attributes of this account. Specify its new attributes.

Address:
0xb8972C13FC2f4Ed90C246D94E7382d38EaF131e6

Attributes

☐ PhysicalNetwork	☐ LogicalNetwork
☑ HMI1	☐ Switch1
☑ PLC1	☑ PLC2
☑ PLC3	☑ PLC4
☑ PLC5	☑ PLC6
☑ PLC7	☑ PLC8
☑ PLC9	☐ Drill1
☐ Drill2	☐ Motor1
☐ Motor2	☐ Motor3
☐ Motor4	☐ Motor5
☐ LightBarrier	☐ BarcodeReader
☐ LoadShuttle	☐ DrillingTable
☐ ChipExtraction	☐ ToolStation

Cancel OK

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-repository[7].

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

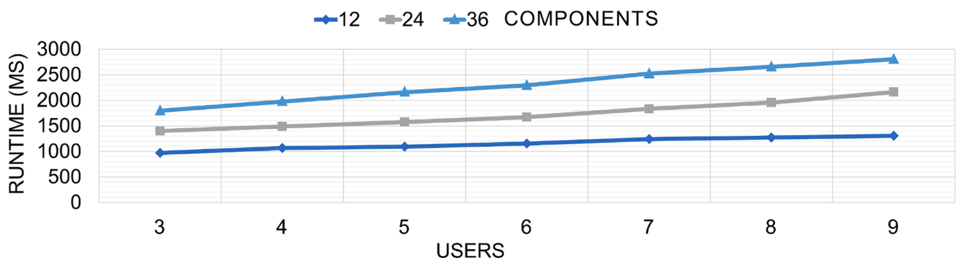


Fig. 9. Runtime values for Algorithm 1 for varying numbers of users and components.

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) project[8]. 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 online[9] 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' ICSs[10] 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 he 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:

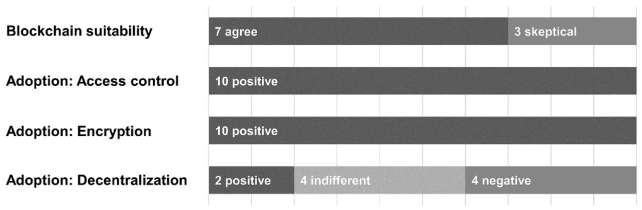


Fig. 10. Expert evaluations regarding the suitability of the presented solution.

- 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 API[11] 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

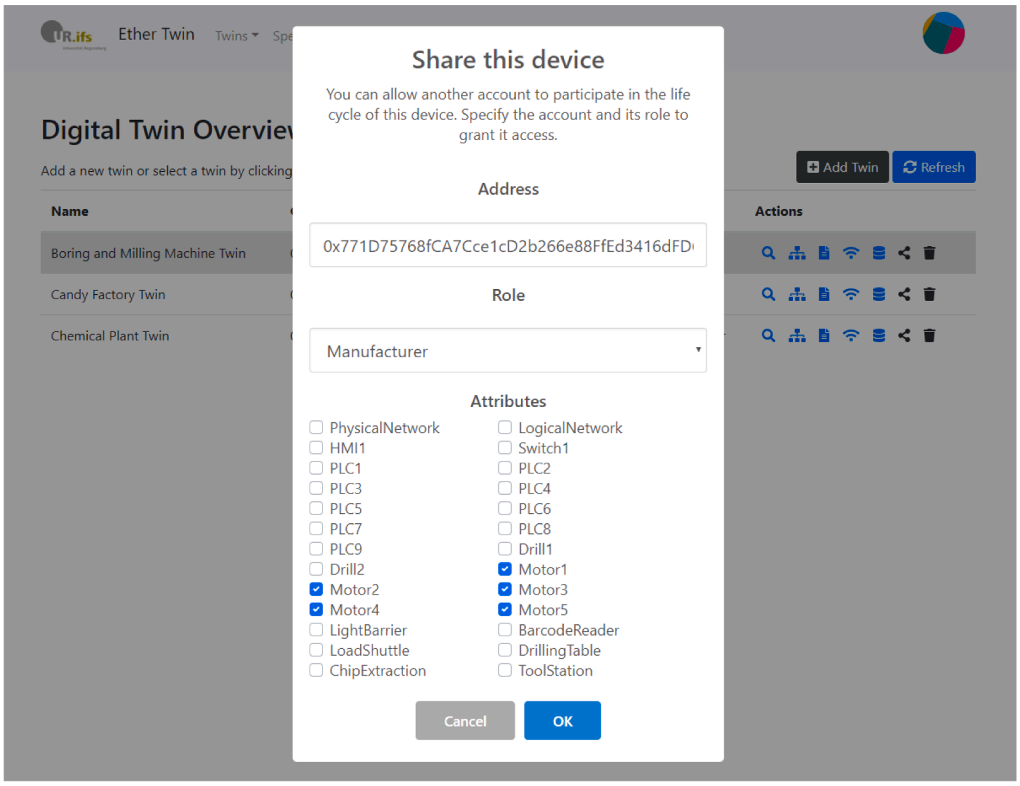
The authors declare that they do not have any financial or nonfinancial conflict of interests

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 project[12], 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



Share this device

You can allow another account to participate in the life cycle of this device. Specify the account and its role to grant it access.

Address: 0x771D75768fCA7Cce1cD2b266e88FfEd3416dFD

Role: Manufacturer

Attributes

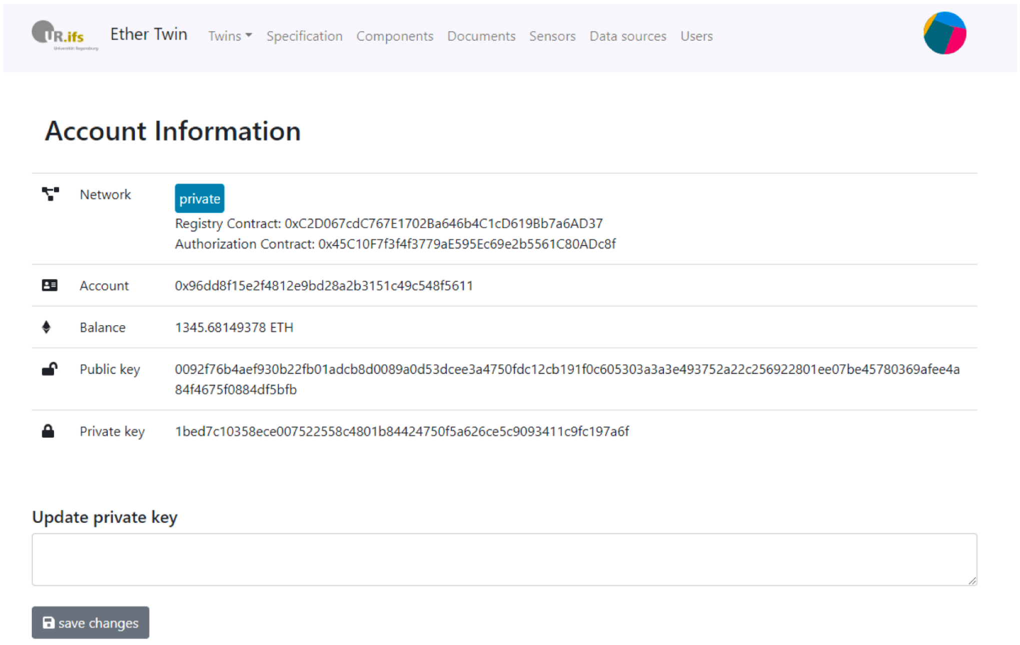
☐ PhysicalNetwork	☐ LogicalNetwork
☐ HMI1	☐ Switch1
☐ PLC1	☐ PLC2
☐ PLC3	☐ PLC4
☐ PLC5	☐ PLC6
☐ PLC7	☐ PLC8
☐ PLC9	☐ Drill1
☐ Drill2	☑ Motor1
☑ Motor2	☑ Motor3
☑ Motor4	☑ Motor5
☐ LightBarrier	☐ BarcodeReader
☐ LoadShuttle	☐ DrillingTable
☐ ChipExtraction	☐ ToolStation

Digital Twin Overview

Add a new twin or select a twin by clicking

Name	Actions
Boring and Milling Machine Twin	
Candy Factory Twin	
Chemical Plant Twin	

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

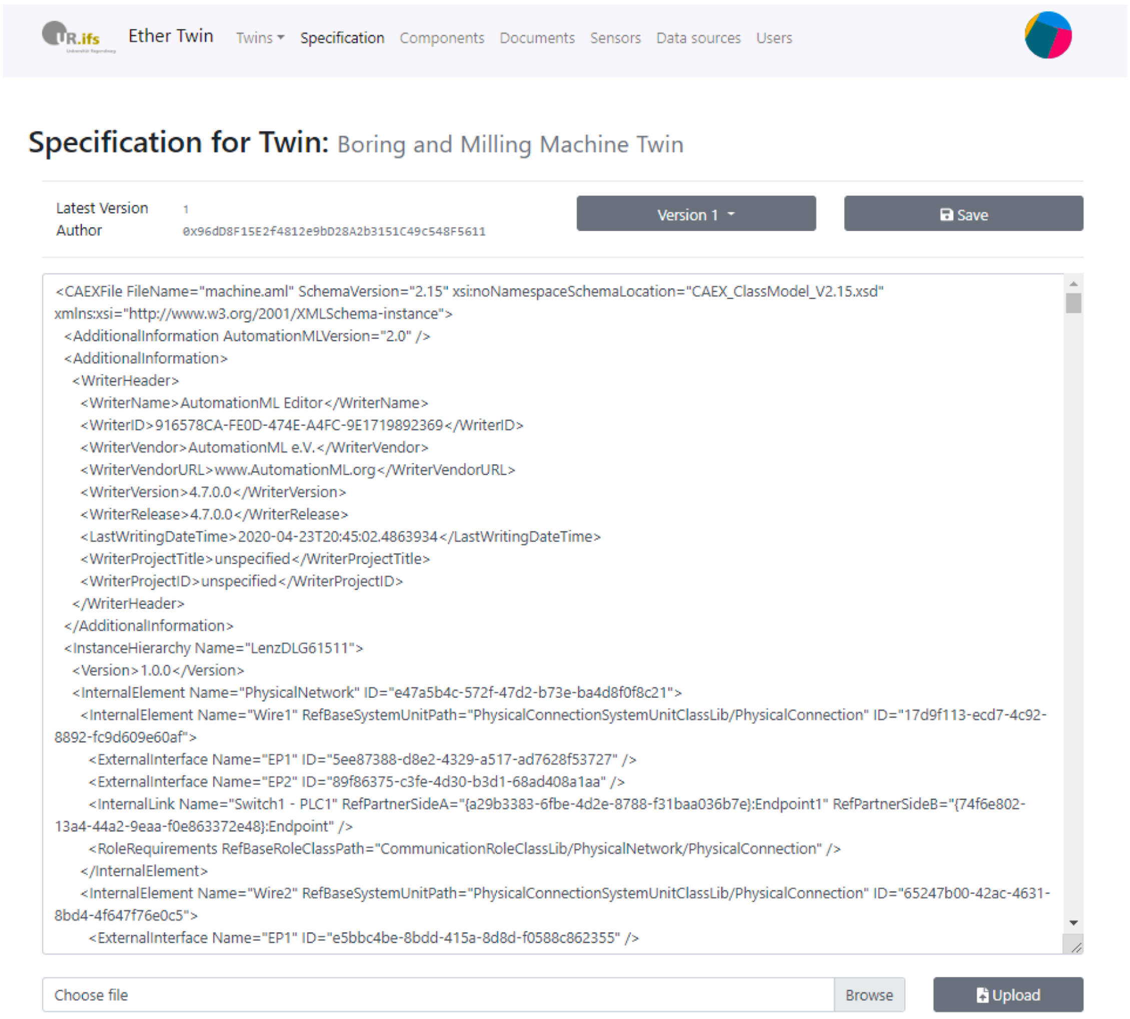


Account Information

Network	private Registry Contract: 0xC2D067cdC767E1702Ba646b4C1cD619Bb7a6AD37 Authorization Contract: 0x45C10F7f3f4f3779aE595Ec69e2b5561C80ADc8f
Account	0x96dd8f15e2f4812e9bd28a2b3151c49c548f5611
Balance	1345.68149378 ETH
Public key	0092f76b4aef930b22fb01adcb8d0089a0d53dcee3a4750fdc12cb191f0c605303a3a3e493752a22c256922801ee07be45780369afee4a84f4675f0884df5bfb
Private key	1bed7c10358ece007522558c4801b84424750f5a626ce5c9093411c9fc197a6f

Update private key

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



Specification for Twin: Boring and Milling Machine Twin

Latest Version 1
Author 0x96dD8F15E2f4812e9bD28A2b3151C49c548F5611

```
<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-9E1719892369</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-fc9d609e60af">
        <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" />
```

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

Data: A set of twins $T = (t_1, \ldots, t_n)$ with mappings for roles M^{UR}_t, permissions M^{PR}_t, attributes M^{UA}_t 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:      $M^{UR}_t \leftarrow getRoleAssignment(t)$
4:      $M^{PR}_t \leftarrow getPermissionAssignment(t)$
5:      $M^{UA}_t \leftarrow getAttributeAssignment(t)$
6:      for each $c \in C_t$ do                                  ▷ generate two symmetric keys $sk$ per component
7:          $(sk^{doc}_{tc}, sk^{sensor}_{tc}) = \mathcal{G}_{enc}$
8:      end for each
9:      for each $u \in (M^{UR}_t \cap M^{UA}_t)$ do              ▷ prepare file keys $fk$ for users
10:         $pk_u \leftarrow$ (DHT) $getUserPublicKey(u)$
11:         for each $c \in C_t$ do
12:             $r \leftarrow M^{UR}_{tu}$
13:             for each $d \in \{doc, sensor\}$ do
14:                 if $(c \equiv a \mid a \in M^{UA}_{tu}) \wedge (p^d_{read} \in M^{PR}_{tr})$ then
15:                     $fk^d_{tcdu} = ECIES\_enc(pk_u, sk^d_{tc})$
16:                     (DHT) $updateFeed(c, fk^d_{tcu})$
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).

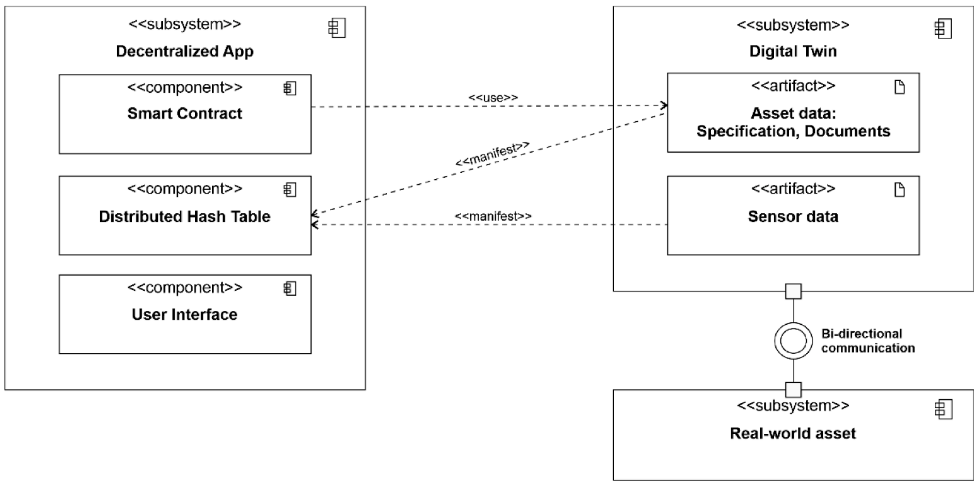


Figure 1: *The concept of DT data sharing and storage realized with a Decentralized App (DApp), with its basis on the blockchain technology.*

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.

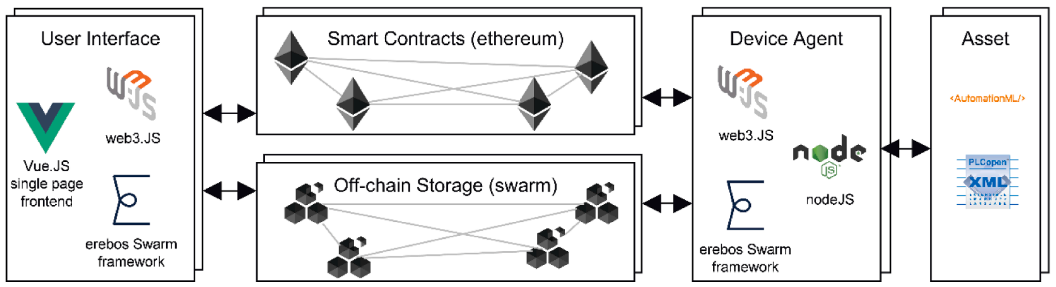


Figure 2: *The selected technologies for the EtherTwin prototype.*

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 tmce* (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 xxxiii* (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'17Proceedings 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. Hosseinian-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. Stjepandic, & 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. Cuppens, L. Wang, N. Cuppens-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 (wetseb)* (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.

1.3 DEALER: Decentralized Incentives for Threat Intelligence Reporting and Exchange [P3]

Status:	published
Publication:	International Journal of Information Security
Submitted:	19 March 2020
Revised:	02 September 2020
Accepted:	11 November 2020
Citation:	Florian Menges, Benedikt Putz, and Günther Pernul. 2020. DEALER: decentralized incentives for threat intelligence reporting and exchange. International Journal of Information Security (2020).

Journal Description: The International Journal of Information Security is an English language periodical on research in information security which offers prompt publication of important technical work, whether theoretical, applicable, or related to implementation.

International Journal of Information Security
https://doi.org/10.1007/s10207-020-00528-1

REGULAR CONTRIBUTION

DEALER: decentralized incentives for threat intelligence reporting and exchange

Florian Menges[1] · Benedikt Putz [1] · Günther Pernul[1]

Abstract
The exchange of threat intelligence information can make a significant contribution to improving IT security in companies and has become increasingly important in recent years. However, such an exchange also entails costs and risks, preventing many companies from participating. In addition, since legal reporting requirements were introduced in various countries, certain requirements must be taken into account in the exchange process. However, existing exchange platforms neither offer incentives to participate in the exchange process, nor fulfill requirements resulting from reporting obligations. With this work, we present a decentralized platform for the exchange of threat intelligence information. The platform supports the fulfillment of legal reporting obligations for security incidents and provides additional incentives for information exchange between the parties involved. We evaluate the platform by implementing it based on the EOS blockchain and IPFS distributed hash table. The prototype and cost measurements demonstrate the feasibility and cost-efficiency of our concept.

Keywords Threat intelligence sharing · Blockchain · Smart contract

1 Introduction

The threat landscape for IT infrastructures has grown steadily in recent years, and this trend is continuing. At the same time, it is becoming apparent that the countermeasures currently available can hardly keep pace with the ongoing attacks. It has been shown that the exchange of threat information is an effective instrument for improving existing countermeasures and the overall situation. It leads to more knowledge about threats, earlier detection of attacks and thus to more effective countermeasures. The potential benefits of the threat information exchange have recently been recognized in the public sector by introducing corresponding legal regulations. For example, several countries already require the reporting of security incidents, especially for critical infrastructure operators.

✉ Florian Menges
Florian.Menges@ur.de

Benedikt Putz
Benedikt.Putz@ur.de

Günther Pernul
Gunther.Pernul@ur.de

[1] University of Regensburg, Universitätsstr. 31, 93053 Regensburg, Germany

While the exchange of threat information offers the aforementioned benefits for the security situation, it can also entail various disadvantages and problems that may prevent companies from participating. These include high additional costs for appropriately trained security personnel and infrastructure, possible data protection problems and the risk of publishing sensitive data. In addition to these problems, a complex set of reporting requirements must be taken into account. Companies must be able to provide non-repudiable proof of accurate reporting, both to avoid penalties and to potentially use the data as evidence in court. Consequently, sustained availability and integrity of the reported data must be ensured. Sharing platforms must address these problems by incorporating legal requirements as part of the design. Additionally, incentive structures must be created for the exchange of threat information, to offset costs and to motivate stakeholders to participate in the long term.

In doing so, we consider two use-cases separately. The platform intends to (1) support the fulfillment of legally **obligatory reporting** and (2) to create economic incentives for **voluntary reporting**. While these scenarios have different requirements and thus follow separate processes, the proposed platform optionally also enables sharing of obligatory reports. Based on these considerations, we formulate the research questions we intend to answer:

Published online: 09 December 2020

- **RQ1**: How can threat intelligence information be exchanged while ensuring availability, integrity and non-repudiation?
- **RQ2**: How can the exchange of threat intelligence information be incentivized?

To solve these problems, we propose a sharing concept and application prototype for a threat intelligence sharing platform based on Distributed Ledger Technology (DLT). DLT provides specific security characteristics, which can differ depending on the blockchain implementation. These usually include *availability*, *integrity* and *non-repudiation* - the three requirements of RQ1 [1]. *Availability* is ensured by the underlying blockchain network, which consists of a large number of geo-distributed nodes maintaining a replicated ledger around the clock. Please note that Proof of Work (PoW)-based blockchains may suffer availability limitations under heavy load [2]. At the same time, *integrity* assurance is provided through a sequentially linked hash chain, which ensures that the current world state is always the result of all past transactions. The consensus protocol assures that state transitions are append-only and previous entries are *non-repudiable*. Distributed Ledgers enable the verifiable decentralized execution of applications in the form of smart contracts, which also provide the option to implement digital currency in the form of blockchain tokens. These tokens can be used to provide decentralized *incentives* by assigning real value to threat intelligence information.

Existing work has attempted to address some of the aforementioned problems using DLT; however, the research questions have not been sufficiently addressed so far (Sects. 2 and 3.3). For this reason, we propose the blockchain-based DEALER platform (**D**ecentralized Inc**E**ntives for Thre**A**t Inte**L**lig**E**nce **R**eporting and Exchange). It fulfills requirements for obligatory Cyber Threat Intelligence (CTI) reporting (Sects. 3.1 and 7.1), while also providing an incentive structure to counteract possible participation drawbacks and to encourage voluntary sharing of CTI. Our contribution includes a novel protocol based on verifiers and token-based incentives to encourage fair sharing of high-quality threat intelligence data. To avoid trusting a third-party platform provider, the architecture is fully decentralized and maintained by independent blockchain operators and the participants themselves. In brief, the platform provides the following key features:

- **availability**, **integrity** and **non-repudiation** as requirements for obligatory reporting
- decentralized **incentives** by leveraging blockchain tokens for purchase and sale of threat intelligence
- transactional **fairness** for both seller and buyer
- **quality assurance** through a verifier system

The remainder of this paper is structured as follows. In Sect. 2, we first provide an overview of approaches for platforms to report threat information, in particular with a focus on meeting the aforementioned security goals. In Sect. 3, we define requirements for the development of our platform. Section 4 introduces our concept for the storage and incentivized exchange of threat intelligence information. In Sect. 5, we propose the system design for the application of our concept and present the implementation of our prototype. The cost structure and thus the practical feasibility of our prototypical implementation are evaluated in Sect. 6. The results of this paper are discussed in Sect. 7, and the paper is concluded in Sect. 8.

2 Related work

The exchange of threat information has been the subject of practical and legislative work in recent years. These include laws in different legislations, such as the NIS Directive[1] in Europe and the IT-Sicherheitsgesetz (BSIG)[2] in Germany, which stipulate the reporting of incidents for providers of critical infrastructures. These legislations are also influenced by data protection requirements, which are, for example, specified by the General Data Protection Regulation (GDPR)[3] or the California Consumer Privacy Act (CCPA).[4] Such regulations have also been addressed in the literature. Schwartz et al. point out fundamental legal aspects of the CTI exchange [3], while Laube and Bhme show that not reporting security incidents may lead to fines in different countries[4]. In addition to this, Bauer et al. have shown in their study on Threat Intelligence Platforms that trust, data integrity, a high platform availability, reporting functionalities as well as data quality are among the key characteristics of CTI platforms[5]. At the same time, the actual exchange of CTI data is already being implemented in practice by various platforms. Examples are IBM X-Force [6] or Facebook threat exchange [7] as commercial platforms as well as MISP [8] and OPENCTI [9] as open source platforms. These platforms allow the exchange of threat information; however, data integrity or availability is not conclusively assured and incentive structures are not available. Central providers can advertise data integrity and availability, but ultimately it is always necessary to rely on the provider to ensure the protection goals are met. This is particularly problematic in the area of possible obligations to provide evidence, as manipulation of the data stock

[1] https://eur-lex.europa.eu/eli/dir/2016/1148/oj.

[2] https://www.gesetze-im-internet.de/bsig_2009/BJNR282110009.html.

[3] https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[4] https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

cannot be ruled out with central providers. At the same time, a single provider usually also represents a single point of failure when it comes to the availability of the platform. Furthermore, existing providers do not yet offer functionalities for quality-assured trading and thus an incentivized exchange of CTI information. In this context, Liu et al. [10] showed that a lack of incentives can even prevent the exchange process from happening. In addition to this, Wagner et al. [11] pointed out the risks that are associated with sharing CTI, which in turn may prevent companies from participating in the exchange, which in the worst case can even lead to the information exchanged being used to attack participants in the exchange.

A great deal of research has been done on the requirements and challenges of implementing CTI platforms. In an early work, Serrano et al. [12] point out the fundamental problems for the exchange of threat information. Dandurand et al. [13] defined requirements for the exchange of information, emphasizing the necessity of assuring data integrity and availability, which is also supported by the work of Brown et al. [14]. Mohaisen et al. pointed out various open research questions in that field, such as possible dangers and negative incentives that may relate to the exchange of CTI [15]. In addition to this, there are also works that deal with specific implementations of CTI platforms, such as the MISP platform by Wagner et al. [8]. However, neither specific integrity or availability requirements nor the integration of incentives is considered. The literature also provides works that address the necessity of creating incentives for the exchange of CTI. Sauerwein et al. conducted an exploratory study that showed a need for incentivizing stakeholders within the exchange process [16]. This work is supported by Sillaber et al. examining the needs of stakeholders and resulting challenges [17]. While these studies provide possible starting points for the use of incentive procedures, the actual use of such procedures within CTI platforms is not considered. Moreover, there are also first approaches that try to implement CTI exchange on decentralized platforms. Alexopoulus et al. present a method for sharing security data streams based on a smart contract and data stream subscriptions [18]. Since the proposed data streams require a direct connection between the parties, the assurance of integrity and availability cannot be guaranteed. Incentive structures are also included in the work, but the design suffers from various weaknesses. Since the described on- and off-chain interactions of buyer and seller are independent of each other, negative consequences for fraud attempts during data transfer can only be implemented to a limited extent. In addition, the quality of the incident can vary during a stream, but only the entire stream can be evaluated by a buyer. This increases search costs on the marketplace because information about alerts is only available in aggregated form. Gong and Lee follow a similar approach with the proposed BLOCIS framework[19]. Here, incident data are also not seen as individual items, but aggregated in threat intelligence feeds. Accordingly, a dedicated assignment and a separate reporting functionality cannot be implemented here. At the same time, the proposed concept only provides automated quality assurance. Homan et al. pursue a different approach by implementing CTI sharing on a private Hyperledger blockchain. This work also shows the potential of blockchains in the CTI sharing area. Although, the possibilities of incentives are briefly described, they are not specifically addressed. Quality assurance and reporting requirements are not considered [20]. The papers shown here also show how important fair and secure exchange is for decentralized platforms. Accordingly, much research has been done in this area in recent years. For example, Shafagh et al. [21] show how homomorphically encrypted data can be exchanged securely. Wagner et al. [22] propose an approach to exchange digital goods based on mediator smart contracts, which enable dispute resolution. This work provides the means for decentralized exchange of CTI with smart contracts, but does not provide a browsable platform or quality assurance.

In summary, it can be stated that different works exist in the area of threat intelligence exchange that consider the requirements and the application of platforms. However, to the best of our knowledge, there is currently no work that allows an incentive-based, fair exchange of CTI information, while maintaining data integrity and availability to comply with regulatory requirements. Accordingly, we briefly summarize the novel points where our solution goes beyond existing work:

- sharing of **individual** incidents on a decentralized marketplace
- **quality assurance** by independent and incentivized verifiers
- support of **legal requirements** for obligatory threat reporting
- token-based **incentives** for voluntary sharing without transaction fees

3 Objective and requirements

The exchange of threat intelligence information can be categorized into two different areas. On the one hand, unidirectional reports of security incidents are stipulated by law and mostly concern companies that are relevant for the functioning of society. On the other hand, bidirectional exchange of security information between companies is done on a voluntary basis. The goal is an improvement of the information basis on security incidents for all participants and to increase their security level. The platform developed in this work aims to cover both use cases by enabling both reporting and exchange of security incidents. We consider the use cases **obligatory reporting** and **incentive-based exchange** of CTI information separately, as they should be independent features on the platform. However, a combination of both approaches should optionally be possible. There are different and unique requirements that result from each of these use cases, which are described in more detail below.

3.1 Requirements for reporting security incidents

The most important requirement for reporting security incidents is compliance with the underlying legal framework. We have taken the German IT Security Act [23] as the basis for our requirements in this matter. In doing so, we first consider the concrete effects of the reporting obligation and derive reporting requirements from this before specifying them in more detail. In principle, existing reporting obligations usually specifically oblige operators of critical infrastructures to report possible outages. The German IT security law explicitly proposes a contact point for the implementation of the reports, which can be used by various parties subject to reporting. This ultimately corresponds to the platform proposed here as a common reporting infrastructure. In terms of content, the law specifically stipulates that failures must be reported immediately and thus places an initial focus on availability. The information should include various technical details as well as information on the respective operator. Within the legislation, a further focus is put on the auditability of critical infrastructures. This shows that ensuring the integrity of the reports is also a key factor in the scope of reporting. In addition, the legislation provides for penalties for failing to report security incidents. Accordingly, it is also important to be able to provide proof that a report has been carried out.

According to this, the first requirement for a functioning reporting infrastructure is that a company must be able to provide incident data and that the legal authorities can obtain these data. Reports on security incidents are to be regarded as time-critical, as the judicial authorities may have to react to reported incidents in good time. For this reason, the provision of a very high **availability** is a key factor in the operation of a reporting infrastructure.

In the context of reports, it is also of utmost importance to be able to prove who submitted a report. On the one hand, this is necessary so that authorities can take the necessary steps to prevent supply bottlenecks, for example. On the other hand, this also provides a guarantee for the reporting institution, as it enables it to prove that the reporting obligation has been fulfilled and thus avoid penalties. This necessity results in the requirement of **non-repudiation** and unambiguous assignment of reports. In addition, a further requirement results from the actual use of the data. Besides being used to prevent damage, the threat intelligence information obtained may also be used as evidence. Specifically, recorded data may either be used as evidence in court proceedings or as proof of damage against contractual partners such as insurance companies. Following this, ensuring data **integrity** is an additional requirement in the reporting process that needs to be taken into account. Besides regulations that stipulate reports of security incidents, there are also regulations regarding the handling of personal data in different jurisdictions, such as the GDPR in the European Union. According to this, the platform must also provide the necessary tools to allow the protection exchanged data in compliance with legal regulations.

3.2 Requirements for an incentive system

In addition to requirements resulting from legislation, there are also functional requirements for exchange platforms. Every exchange of information on security incidents is accompanied by various risks. When publishing information, companies risk to accidentally leak important data. This may, for example, include company secrets or information about the company infrastructure that may, for example, simplify attacks on that company. In addition, a reporting process involves costs for the collection, processing and dissemination of incident data. At the same time, the benefits of participating in an exchange platform are often difficult to quantify, especially with comparatively low legal penalties for omitted reports. From these points it can be concluded that companies tend to have little intrinsic motivation to report incidents themselves, whereas the motivation to passively obtain information from a reporting platform is likely to be high. As a result, an **incentive system** that motivates every participant on such a platform to actively participate can be defined as a further essential requirement for the sustainable functioning of such a platform (**RQ2**).

3.3 Platform comparison

As shown above, several other platforms already exist that enable the exchange of threat intelligence information. Among them are both centralized and decentralized concepts covering different use cases. This leads to the problem that platforms offer different features and each of them offers its own approach to addressing protection targets. In order to demonstrate the advantages of the DEALER platform compared to existing concepts this section compares the DEALER platform to Facebook Threat Exchange (FB-TX), IBM X-Force, MISP, OpcenCTI and Trident as presented in Sect. 2. Specifically, the key features of DEALER for the creation of incentives as well as the implementation of reporting obligations are compared individually for all platforms. More specifically, the aforementioned protection goals of availability, integrity, non-repudiation, fairness, quality and

Table 1 Comparison of CTI sharing platforms

	FB-TX	X-Force	MISP	OpenCTI	Trident	DEALER
Platform availability	◐	●	◐	◐	●	●
Data availability	●	●	●	●	○	◐
Integrity	○	○	○	○	◐	●
Non-repudiation	○	○	○	◐	●	●
Incentives	○	○	○	○	●	●
Fairness	○	○	○	○	◐	●
Quality assurance	○	○	○	○	◐	◐

○ Not addressed, ◐ insufficiently addressed, ● explicitly addressed

the possibility of creating incentives for exchange are considered. Depending on the use case of the notification or exchange, the availability of the platform itself as well as the availability of the data in the specific case are also distinguished for the determination of the availability protection goal. This separation is introduced since one problem within decentralized platforms is to keep exchangeable data available for trade at all times. The full comparison made here is shown in Table 1 and is explained in more detail below. The comparison rates different platform protection goals with values from "not addressed" through "insufficiently addressed" to "explicitly addressed". Significant differences between centralized and decentralized platforms are apparent at first glance in Table 1. In the following, these are broken down in more detail once again.

Platform availability. In the FB-TX, MISP and OpenCTI platforms, platform availability is not given special consideration. No specific statement can be made for MISP, as it is operated independently by different communities. Moreover, FB-TX already had several outages in Q2 2020. [5] Accordingly, no increased availability can be assumed for these platforms. X-Force addresses this problem using increased parallelization, however, outages regularly occur here as well. Trident and DEALER, on the other hand, are operated on decentralized blockchains ETH and EOS, on which outages are very rare due to the high number of network nodes.

Data availability. This property is naturally very high for all central platforms, as data can be uploaded and is available regardless of the status of the user. In contrast, Trident is dependent on the availability of the user and has a correspondingly low data availability. Despite the decentralized approach, DEALER tries to address this problem by sharing the load between several users. This is explained in more detail in Chapter 4.3.

Integrity. The data integrity is not considered in any of the centralized approaches whereas Trident and DEALER implement them. While this is only partially addressed in Trident as the data exchange is based on a direct stream, the DEALER concept provides for a complete integrity assurance of the data. At the same time, data integrity is a characteristic that can be ensured particularly well by decentralized platforms without the need for trust.

Non repudiation. This property is only addressed in OpenCTI, Trident and DEALER. With OpenCTI, however, this is only partially the case, as this is ensured by the platform administrator and a corresponding level of trust is required. **Incentives:** Both Trident and Dealer offer the possibility of evaluating incident data and exchanging it via a marketplace. Such incentive mechanisms are not provided for in the central platforms.

Fairness. Within the DEALER platform exchange fairness is specifically addressed. Trident also addresses this, however, only peripherally by creating a relationship of trust between the participants. On the central platforms this problem is currently not considered at all.

Quality assurance. This property is only addressed by the decentralized approaches so far. Central platforms do not yet take this into account. However, MISP names quality assurance as an important goal for Future Work.

Overall, this comparison shows that the implementation of a decentralized CTI exchange platform can create various advantages for the exchange. These include features such as the creation of incentives, quality assurance and the integration of fairness mechanisms. These were not implemented on existing platforms although there are no significant technical obstacles. On the other hand, they also include criteria such as platform availability, integrity assurance and non-repudiation. Due to the inherent characteristics of decentralized platforms, these can be mapped very well and without the need of ensuring trust. Although the concept of the DEALER platform does not fully meet all criteria, it is clear that such a system is superior to traditional approaches in many respects.

3.4 Shared CTI data

In general, any data format can be used for the exchange within the platform. Within this work as well as within the development of the platform, we have used the state-of-

[5] https://developers.facebook.com/status/dashboard.

the-art data format STIX2 for the exchange as well as the reporting of incidents.

```
{
  {
    "type": "bundle",
    "id": "bundle--2a25c3c8-5d88-4ae9-862a-
        cc3396442317",
    "objects": [
    {
       "type": "indicator",
       "id": "indicator--1",
       "created": "2014-02-20T09:16",
       "name": "File hash for Poison Ivy
           variant",
       "description": "This file hash
           indicates that a sample of Poison
            Ivy is present.",
       "indicator_types": [
           "malicious-activity"
       ],
       "pattern": "[file:hashes.'SHA-256' =
           '9ef537f25c895bfa7825...']",
       "valid_from": "2014-02-20T09:00"
     },
     {
       "type": "malware",
       "id": "malware--2",
       "created": "2014-02-20T09:16",
       "name": "Poison Ivy",
       "malware_types": [
           "remote-access-trojan"
       ],
       "is_family": false
     },
     {
       "type": "relationship",
       "id": "relationship--3",
       "created": "2020-02-29T18:09",
       "relationship_type": "indicates",
       "source_ref": "indicator--1",
       "target_ref": "malware--2"
     }
    ]
}
}
```
Listing 1 Exemplary STIX 2 bundle

Listing 1 shows a shortened excerpt of such an exemplary STIX2 data packet, which is basically provided in the JSON data format. Such a data packet is always enclosed by the structuring unit *bundle* which allows a unique assignment of the data packet. Such a bundle contains the various STIX objects and references between these objects. The shown example contains with the object "indicator-1" information about an indicator, i.e. a pattern that indicates a possible security incident. Furthermore, the object "malware-2" contains information about the detected Malware "Poison Ivy". Finally, a connection between the two objects is established by the relationship object "relationship-3", which references both objects. In addition to an impression of the basic syntax and design of a security incident with STIX2, this example also gives an insight into the dynamic data model of STIX2. A bundle object can contain any number of STIX objects, which in turn can be dynamically connected to each other by means of relationship objects.

4 The DEALER sharing concept

In this section we present the DEALER concept, which is designed to fulfill the previously defined requirements and to provide an incentives structure for sharing CTI information. This includes an ecosystem describing the stakeholders in the system, their roles and relationships and a marketplace describing the processes and concepts within the ecosystem, designed to guarantee sustainable CTI exchange. This section provides an overview of the relationships within the system and the overall idea of the concept. The individual processes within the system are described subsequently.

The entire system, which is outlined in Fig. 1 consists of five essential components. At the center of the system is a **blockchain** and a **distributed database**. These form the technological basis for the implementation of smart contracts, integrity-secured storage of exchange processes and provide decentralized storage structures for reported security incidents. The starting point for reports within this system is **Critical Infrastructure Compounds**. These include the critical infrastructure operator, an IT service provider if applicable, and a CTI provider. The CTI provider takes care of external communication and acts as a so-called contact point, a construct that can be derived from legal requirements for incident reporting. The information collected is intended for either Associated Institutions or Organizations. **Associated Institutions** describe participants who are interested in the reported information within the scope of reporting obligations. These can, for example, be legal authorities to which a reporting obligation exists. These can also be other institutions, such as insurance companies, to which a possible claim can be made accessible via the platform. On the other hand, there are **Organizations** that are not affected by reporting obligations, but are nevertheless interested in participating, for example to increase their own level of protection. Analyses and services within the system are provided by the **CTI ecosystem**. This enables external service providers to bring their services into the system. For example, verification providers can offer qualitative incident data evaluation, or analytics providers can aggregate information on several incidents and offer it within the system.

DEALER's overall concept defines two central use cases: statutory incident reporting and incentive-based threat intelligence exchange. Both concepts are briefly described below before we take a closer look at the underlying processes.

Obligatory reports are generated by the Critical Infrastructure Compound and transferred to the blockchain. The transmitted data are pseudonymized and encrypted in such a

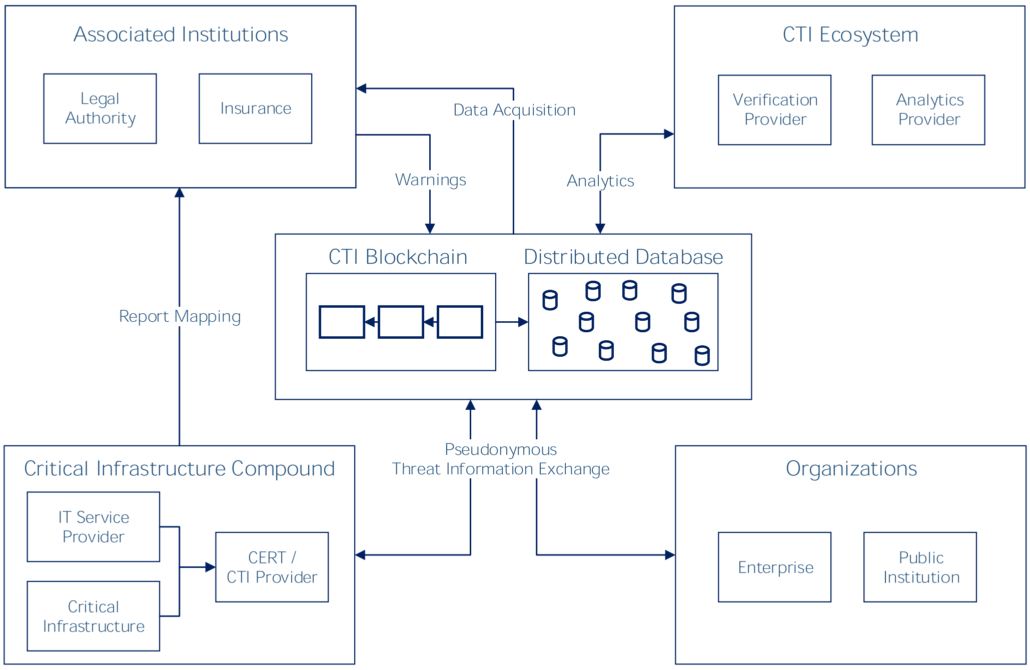


Fig. 1 High level overview of the DEALER threat intelligence sharing concept

way that only the receiving authority can access it. In connection with such a report, the data can also be made available to other users of the platform as part of the incentive-based exchange. However, this step is explicitly optional and must be actively selected.

The **incentive-based exchange** process is based on an economic model, where participants can offer and demand information on security incidents. For this purpose, a separate token is introduced on the platform, which functions as an internal currency and is used as economic reward for active participants. When threat information is provisioned, structured incident data are transferred to the blockchain in encrypted and pseudonymized form. The information provided can then be sold to other participants or made available as a report. The uploaded incident information is assigned to verifiers who ensure its data quality against a fee. After successful verification, the data can be traded on the platform at the previously defined price.

In addition to these two sharing mechanisms, legal authorities may additionally issue global warnings regarding threats to all participants. In some legislations, such as the IT security law in Germany [23], such global warnings are part of the reporting obligation and thus necessary for compliance. The warnings also represent an additional benefit for the platform participants: the free CTI provided by the legal authority supplements purchasable incident information.

After this high-level introduction to the basic concept of DEALER, the core processes of the platform are presented in more detail below. They include Registration (4.1), Sharing (4.2), Verification (4.3), Purchase (4.4) and Fairness (4.5).

4.1 Registration

Initially, participants must register to be able to transact on the decentralized marketplace. Each participant has an account with a balance of fungible tokens, which may be used to trade incidents. To prevent sybil attacks, we require a fixed initial token stake s_i to create the participant's balance. This prepayment requires a meaningful investment, while not deterring new users. The user balance is managed by the platform. Withdrawals are allowed on request up to the initial fee, which must remain until the participant closes the account.

Verifiers are treated separately during registration, as they are given free access to incident information and must evaluate it. The purpose of registration is to achieve a unique identification of the verifier, for example by requesting a tax number, identity documents or a social security number. This registration process is intended to prevent the risk of verifier misuse (i.e. free-riding or submitting default ratings). In contrast to regular participants of the platform, verifiers must be approved before participating in the verification process. During bootstrapping of the verifier pool, approval can be conducted by the platform developer. Once the verifier pool has reached the minimum size (Sect. 7.3), new participants can be approved through majority votes of existing participants.

Additionally, the platform provides an exclusion option for malicious verifiers. Exclusion of a verifier must be approved by a majority of the verifier pool through multisignature votes. Any verifier may initiate such a vote by providing evidence for several instances of misbehavior (i.e. repeatedly submitting default or unrealistic verification reports).

Besides preventing misuse, the goal of authenticating verifiers is to ensure an intrinsic interest in the analysis of security incidents and possession of the necessary technical expertise for actual incident information assessments. Appropriate verifiers could, for example, be threat intelligence vendors, CERTs or security operations professionals.

4.2 Sharing

Figure 2 shows a BPMN diagram of the sharing process from incident detection to data upload, verification and provisioning on the platform. Initially, the participant locally performs required preprocessing steps. These include anonymization (removing private data and identifying details), addition of public descriptive metadata and encryption of the incident with a symmetric key k. The metadata also include a sale price p_s. A signed transaction is submitted to the platform and the incident is uploaded to the distributed database. If the participant decides to sell the incident to other users, a verification fee p_v must be paid once with the initial transaction. We suggest $p_v \sim 0.6p_s$ to reward verifiers depending on the value of the incident. The incident is then made available on the marketplace and verification is initiated. Three random verifiers are chosen from the verifier pool. The seller then uploads three keys $k_{v1}/k_{v2}/k_{v3}$ for each chosen verifier, encrypted with each verifier's public key, and notifies the platform at time T_1. The verifiers retrieve and decrypt the uploaded incident with their individual key file. They assign an initial rating value based on a set of platform-provided quality metrics (Sect. 4.3).

The verifiers submit the verification result to the platform. If all results arrive until time T_2, the verification fee p_v is distributed equally among the verifiers ($\frac{p_v}{3} = 0.2p_s$ per verifier, as noted above). If any verifier does not respond, the seller may trigger a replacement of inactive verifiers. These verifiers must respond until time T_3 ($T_3 > T_2 > T_1$), else the seller may request a removal of the incident from the platform and partial reimbursement of the verification fee ($\frac{p_v}{3}$ per missing verifier).

For obligated incident reporting, the participant may want to keep the incident confidential and not share it with verifiers. In this case, the participant only uploads a key for the regulatory authority and no verification is performed. The platform provides a timestamp and proof of reporting for the incident.

4.3 Verification

The data quality verification conducted by verifiers serves as an incident reputation bootstrapping mechanism. We propose a 5-point rating scale for incident quality from 1 (very low) to 5 (great). The verification needs to be as objective and meaningful as possible to provide guidance for buyers, since the actual data are encrypted. The following items serve as verification guidelines:

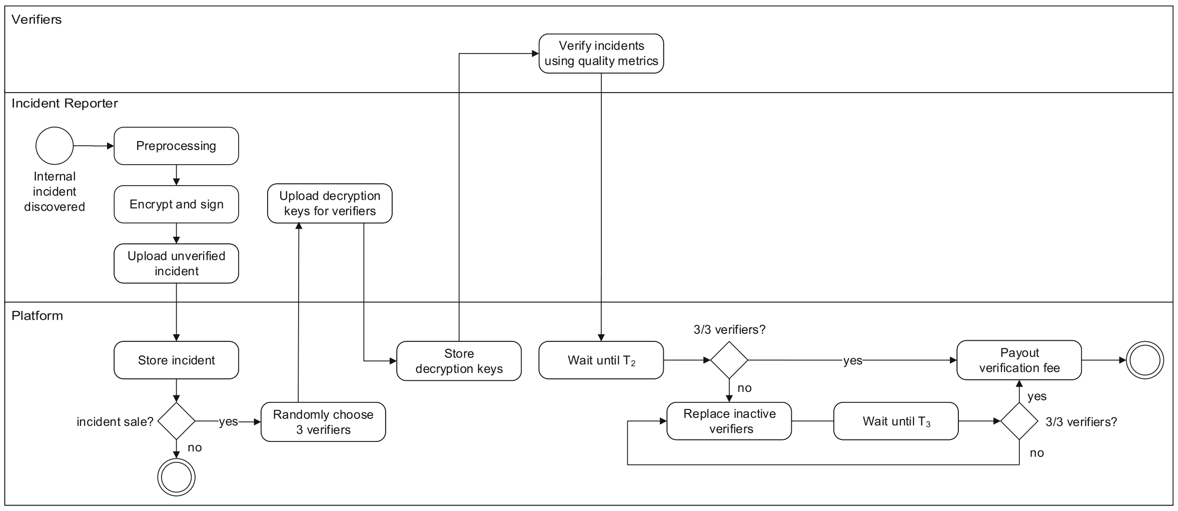


Fig. 2 Incident sharing process on the DEALER platform

- consistency with metadata of the seller's previous incidents
- similarity check for incident metadata and verified incidents
- assessment of various threat intelligence quality indicators [24]

After receiving the incident data, each verifier independently performs a verification of the contained information. A basic consistency check using metadata of the seller's previous incidents verifies that the incident originates from the same industry. To avoid duplicates and resold incidents (Sect. 7), verifiers compute a similarity score to other previously downloaded incidents (i.e. using the *simhash* algorithm [25]). For apparent duplicates, verifiers then submit a low score without additional quality assessment.

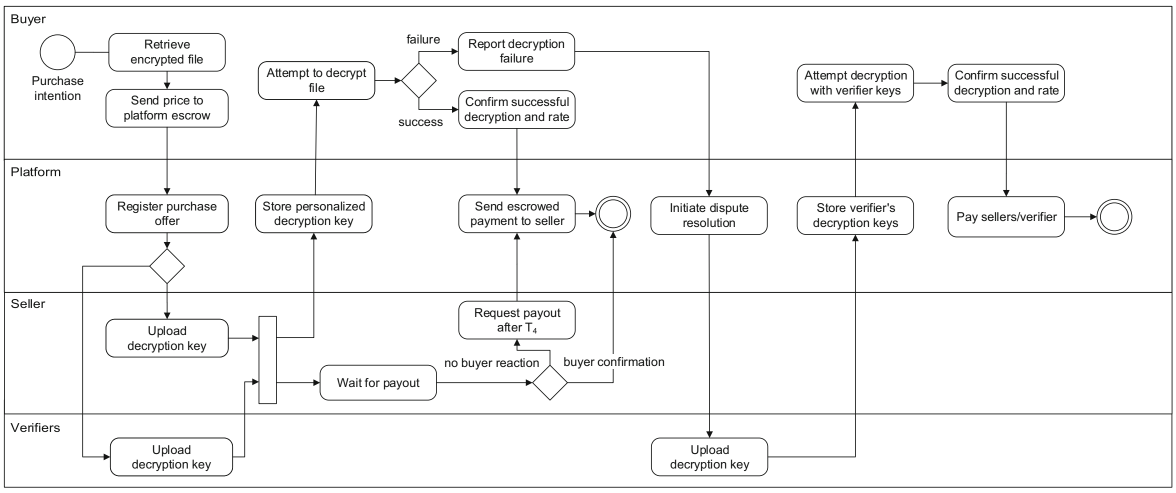
Regarding threat intelligence quality indicators, the platform provides a structured assessment process. This procedure is intended to help verifiers make objective and comparable assessments of security incidents by iteratively processing predefined questions.

To achieve this, the implemented questions are based on objective CTI data quality indicators developed for STIX2 [24]. The quality criteria are divided into three major domains. These include information about the contained data, object representations within the data and the completeness of the available information. In particular, the data model domain reflects information about the *representational consistency* of the data representations and the *concise representation* of the stored information. The object metrics area considers the *objectivity* of the data collected as well as metrics about the *relevancy* of the stored data regarding the situation described. The third domain addresses the completeness of the available information in more detail. This includes the examination whether an *appropriate amount of data* is used to convey the facts presented. In addition to this, the *syntactic accuracy* of the data transported as well as the *schema completeness* of the data is checked.

4.4 Purchase

The incident purchase in Fig. 3 process starts off with a potential buyer browsing the repository of previously uploaded incidents. For this purpose the platform front end offers sophisticated search and filter functionality. Metadata and ratings are provided for each individual incident by verifiers and past purchases. Once an incident of interest has been identified for purchase, the buyer retrieves the encrypted incident to verify its availability. If the incident is available, the buyer places an order for the incident and pays tokens corresponding to the sale price p_s to the platform escrow. After the order has been placed successfully, the key for decrypting the data record is released in the next step. In order to speed up this procedure and not to have to wait for the presence of the seller this can be done either by the seller or by the verifiers. This is possible because all verifiers involved also possess a valid key k_1, k_2 or k_3, as shown in Sect. 4.2. For successful purchases, the key is automatically issued in the background where the decryption key is encrypted with the public key of the buyer k_b and uploaded. In case of successful decryption, the buyer notifies the platform by sending a confirmation along with an incident rating.

If the decryption fails, the buyer notifies the platform about the failure, which initiates the dispute resolution process. Any verifier may then provide an independent copy of the decryption key to the buyer. In the unlikely event that the buyer is



Buyer
Purchase intention
Retrieve encrypted file
Send price to platform escrow
Attempt to decrypt file
failure
success
Report decryption failure
Confirm successful decryption and rate
Attempt decryption with verifier keys
Confirm successful decryption and rate
Platform
Register purchase offer
Store personalized decryption key
Send escrowed payment to seller
Initiate dispute resolution
Store verifier's decryption keys
Pay sellers/verifier
Seller
Upload decryption key
Wait for payout
Request payout after T4
no buyer reaction
buyer confirmation
Verifiers
Upload decryption key
Upload decryption key

Fig. 3 Purchasing process on the DEALER platform

still unable to decrypt the file, keys must be uploaded by additional verifiers to resolve the dispute. Once the buyer is able to decrypt the file, the buyer submits a rating for the incident and closes the dispute. For providing decryption keys, contributing verifiers receive an equal share of the dispute fee p_d as an incentive. The dispute fee is deducted from the sale price and should be proportionately low for two reasons. First, by monitoring the blockchain for disputes verifiers can upload key copies in an automated fashion, requiring little effort and thus little incentive. Second, sellers should not lose a large amount of the sale price in case of unwarranted disputes caused by dishonest buyers. However, dishonest sellers should still be punished to encourage honest incident sale. We thus suggest an initial value of $p_d \sim 0.10 p_s$, subject to further practical evaluation.

A time lock T_4 is in place to allow parties to redeem their tokens if the counterparty fails to respond. If the buyer does not report decryption success or dispute, the seller may collect the sale price after T_4 has expired. If the seller never accepts the offer, the buyer may redeem the locked tokens after T_4.

4.5 Fairness

Fairness of incident purchase must be considered from two perspectives:

– **Seller fairness**: An honest seller is guaranteed to receive the advertised sale price for providing a correct decryption key.
– **Buyer fairness**: An honest buyer is guaranteed to receive the plaintext of the purchased incident, or is refunded the deposited purchase price.

We guarantee Fairness based on the following assumption: There is always at least one honest verifier that provides a valid decryption key. After verification, there are at least four copies of the decryption key (the seller and three verifiers) available on the platform. It is reasonable to assume that there is at least one honest participant among these four, which provides a decryption key in case of an issue with the seller's key.

We now analyze the various ways how seller and buyer may attempt to cheat, and how the protocol mitigates these attempts.

Buyer fairness. The honest verifier assumption means that the buyer will always receive a decryption key, and that there is no scenario where the buyer will not be able to decrypt the file. Conversely, the buyer will also not receive the deposited price back. In case the seller attempts to cheat by uploading a wrong decryption key for the buyer, the buyer can initiate a dispute to receive a correct key from a verifier. Verifiers receive a dispute fee p_d as participation reward for uploading correct keys during a dispute. The seller is thus disincentivized to send wrong keys, since that increases the likelihood of a dispute and results in a loss of p_d tokens.

In case both seller and verifier keys are incorrect, the buyer may be unable to decrypt the item at all. This will not occur in practice based on the assumption that the majority of verifiers is honest and provides correct keys. This assumption can be made based on two properties of our platform:

1. random assignment of verifiers to incidents makes seller-verifier pairings unlikely, and repeated collusive arrangements are time-consuming
2. misbehavior is disincentivized through significant verifier registration requirements (Sect. 4.1) coupled with the possibility of exclusion

We have thus ensured that the seller is punished for uploading wrong key material, while the buyer is able to decrypt the purchased file. To increase the buyer's confidence in receiving a correct key, the time of last platform activity of an incident's verifiers can be shown in the user interface.

Seller fairness. The buyer may attempt to cheat the seller by not responding after the seller has provided the decryption key. For this reason, there is a deadline for the buyer to respond, which starts from the time the seller has uploaded the key and ends after time T_4. If there is no response after expiry, the seller may redeem the purchase price.

The buyer may also collude with the verifiers to falsely vote for seller misbehavior. In this case the honest seller would lose out on p_d tokens deducted from the sale price. This scenario is unlikely, since the buyer has no incentive to collude with verifier. If buyer and verifier are in contact, they could exchange data and tokens through another channel with a reduced price. In practice, this is unlikely to occur, since there is a large overhead for buyers to contact verifiers for every incident they are interested in.

If not colluding with a verifier, the buyer has no incentive to blame the seller. He cannot receive any tokens back that were paid for the sale, and he is guaranteed to receive a correct decryption key if at least one verifier is active.

These considerations guarantee Seller Fairness, with the restriction that the seller may lose out on a small portion of the sale price p_d in case of a dispute. Disputes cannot be prevented by the seller, but buyers have no incentive to start disputes, so we expect them to be negligible in practice.

5 Application prototype

To implement the sharing concept, we choose a combination of blockchain technology and distributed hash tables. This avoids having to trust a single third-party service provider to provide storage and confidentiality. A data storage distributed in this way can be maintained collaboratively and only by participants interested in sharing data. Blockchain networks also allow utilizing virtual currencies that provide possibilities to realize built-in sharing incentives for participants. In the following we first discuss the technologies used for our prototype (5.1). Subsequently, we develop the conceptual architecture (5.2) and briefly present our prototypical implementation of the sharing platform (5.3).

5.1 Technology selection

In this section we will first discuss the underlying technologies for our sharing platform. This includes the permission model, the approach for storing incident data as well as the chosen blockchain platform.

Blockchain platform. The first consideration when deciding on a blockchain platform is the choice between a permissioned network and a permissionless public blockchain. *Permissioned* networks consist of a fixed set of participants that each operate a node of the private network. We experimented with the permissioned blockchain Hyperledger Fabric, but found many obstacles during our research that made it unsuitable for the DEALER platform. These include missing native token support, no means to exchange tokens for fiat currency, and the increased barrier to entry caused by the need to deploy and operate a private Hyperledger Fabric node. The latter results in high initial costs and maintenance costs for updates and monitoring, while availability is less certain due to the limited number of blockchain nodes. *Permissionless* blockchains are operated by independent miners that are incentivized through mining rewards distributed by the consensus protocol (i.e. Proof of Work or Proof of Stake). The blockchain infrastructure is thus already available, but transaction fees must be paid to the maintainers of the platform. Public blockchains also provide a high number of distributed nodes that guarantee high availability, while token distribution can be handled transparently using existing exchanges. Since high availability and incentives for participants are essential aspects of our concept, we choose a *permissionless* blockchain approach for our concept.

Commonly, researchers use Ethereum for permissionless blockchain application prototypes due to its good tool support and large developer community [26]. Unfortunately, the intermittently high transaction costs[6] represent a barrier to entry and reduce the ability to provide incentives for participants. The low maximum transaction throughput of around 15 transactions/second [27] amplifies this issue, as transaction fees rise when the network is congested. This problem is exacerbated when transaction demand increases to extreme levels [28]. Therefore, after evaluating both permissionless and permissioned blockchains, we settle on the EOS blockchain[7] for our implementation. We utilize EOS as opposed to other permissionless blockchain platforms like Ethereum for several reasons. First and foremost, EOS does not charge users transaction costs. Transaction allowances are determined based on staked EOS tokens, thus lowering the long-term cost of using the platform. In addition, EOS provides more scalability regarding transaction throughput (up to 8,000 transactions/second [29]). The EOS network itself is maintained by hundreds of nodes around the world using delegated Proof of Stake (dPoS) consensus. 21 active block producers are selected from a list of candidates[8] based on the votes of EOS token holders. The block producers themselves are encouraged to participate in the network through block rewards (EOS token), which they receive for creating new blocks. Other nodes serve as standby nodes and store a copy of the blockchain, ready to assist if an active producers goes offline or no longer has enough votes. Since the 21 active producers run a deterministic byzantine fault-tolerant protocol among each other, at least 15 colluding producers are required to take over the blockchain.

Data storage. Due to high costs associated with smart contract data storage, larger data items are commonly stored off-chain in blockchain applications [30]. One way of trading data using blockchain is settling the trade on-chain and trading the actual data off-chain [18]. This avoids the need for another storage platform besides the blockchain. However, it also requires the seller to re-upload data to every buyer, which means that both seller and buyer need to be online at the same time. A decentralized off-chain storage platform avoids this issue. To ensure an integrity link between the blockchain network and the off-chain store, the database should be content-addressable. Since only encrypted information is stored off-chain, access control is not required. Distributed Hash Tables (DHTs) provide these properties: they offer public, distributed and content-addressable key-value data storage. We opted for IPFS[9] as the DHT implementation in the prototype. IPFS is widely used in research as an off-chain storage solution, and it provides the features needed for sharing CTI data and encryption keys.

In the DEALER prototype, each participant operates a IPFS node. IPFS nodes are simple to set up; after installation only a single command is required to start the daemon. We use these IPFS nodes to obtain fixed address for each peer for sharing dynamic content, referred to as its *IPNS address*. The node's IPNS address is based on the hash of the peer's public key and can only be updated with a signed update from that peer. We exploit this functionality to statically address each user's shared incidents and decryption keys. We leverage the

[6] https://bitinfocharts.com/comparison/ethereum-transactionfees.html.

[7] https://eos.io.

[8] https://bloks.io.

[9] https://ipfs.io.

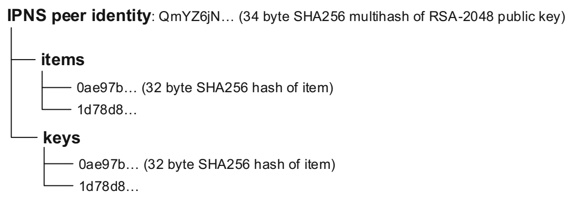

IPNS peer identity: QmYZ6jN… (34 byte SHA256 multihash of RSA-2048 public key)
items
0ae97b… (32 byte SHA256 hash of item)
1d78d8…
keys
0ae97b… (32 byte SHA256 hash of item)
1d78d8…

Fig. 4 IPFS off-chain storage folder hierarchy (for each user)

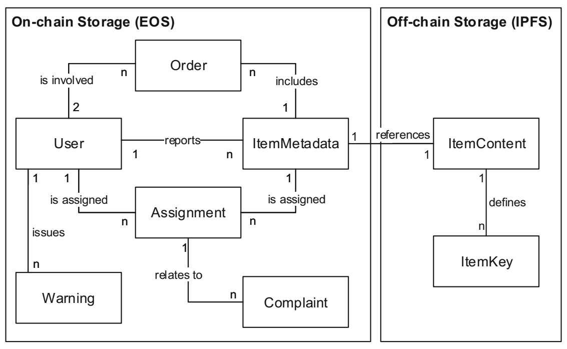


Fig. 5 Simplified entity relationship model of data stored on the DEALER platform

IPFS Mutable File System to create a local folder hierarchy corresponding to the files we intend to share (Fig. 4). The root hash of this folder hierarchy changes every time an item or key is added to a folder. Each time that happens, the updated hash is published to the peer's IPNS address. Other peers can resolve this address to retrieve the latest incidents and keys shared by other users. By *pinning* content hashes, verifiers permanently replicate the encrypted incident shared by the seller to ensure its availability. Verifiers are incentivized to replicate seller content, since they potentially profit from each sale in case of a dispute (Sect. 4.5).

5.2 Architecture and data model

As shown in Fig. 1, the prototype architecture consists of a smart contract on the EOS blockchain platform and IPFS-based decentralized storage. The blockchain platform provides executable smart contracts that implement the *Platform* role in the processes described in Sect. 4. IPFS provides storage capabilities for reported incident data and encryption keys. It also provides pseudonymous identity: Participants sign up with blockchain accounts, which are authorized through public-private key pairs and represented by unique addresses. Figure 5 gives an overview of the platform's data model.

The model shows a distinction between *on-chain* and *off-chain* storage. The on-chain storage manages transaction information and metadata including assignments of users (*User*), reports (*ItemMetadata*), votes *Assignment* and purchases (*Order*) of incident data. The off-chain storage holds the actual incident data (*ItemContent*) as well as the encrypted decryption keys for the information (*ItemKey*). The *ItemMetadata* table contains the reported incidents' metadata, including a short description, the originating industry, the price and a reference to the reporting user. *ItemMetadata* also contains the CTI item's hash, which links the metadata to the full incident data *ItemContent* stored off-chain. Using the hash reference, data can be retrieved from IPFS through a DHT lookup and verification of the retrieved file against its hash reference. The assignment of randomly selected verifiers is done using the *Assignment* table by establishing a link between the verifying user and the respective item. This table also stores the results of item verification, while cumulative results of verification and rating processes are stored in the *ItemMetadata* table. The assignment table is additionally linked to the *Complaint* table, which stores complaints about inaccurate verifications. The *Order* table finally contains the transactions associated to an order, where a transaction establishes the relationship between the buyer and the seller, as well as the item concerned. Besides storing report items, the application also allows the issuance of warnings. These can be inserted by authorities as a specific type of user and stored in the table *Warning* on chain.

5.3 Application prototype

The prototypical implementation of the platform consists of three major components: the smart contract on the EOS blockchain based on EOS C++ code, the IPFS data storage and a DApp (Decentralized Application) front end based on Node.JS. Since Smart Contract and data storage were already described previously, this section focuses on the implementation of the DApp.

Figure 6 shows the implemented components (node.JS server and EOS smart contract) and their interactions with the distributed system. Each DEALER participant runs a node.JS server which manages the encryption keys and blockchain wallet for the organization. It also serves the web interface to internal users. On user requests, the node.JS server interacts with the IPFS network and the EOS test network. CTI data and file keys are stored at the local IPFS node and managed through its IFPS identity. Requests for new CTI data are resolved through the IPFS network. Blockchain transactions are sent to the smart contract on the EOS test network, and data are read back through the EOS node's HTTP API.

Figure 7 shows the user interface of the DApp. The application's user interface offers four fundamental areas tailored to each participant type. The area *BUY* allows potential buyers to get an overview of offers on the platform and to buy and download available incident information. The overview contains a short description of the incident information as well

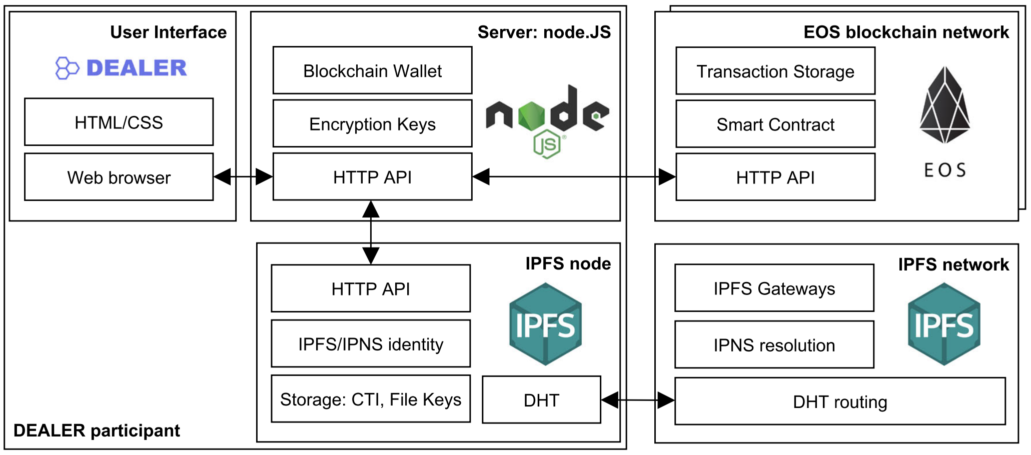
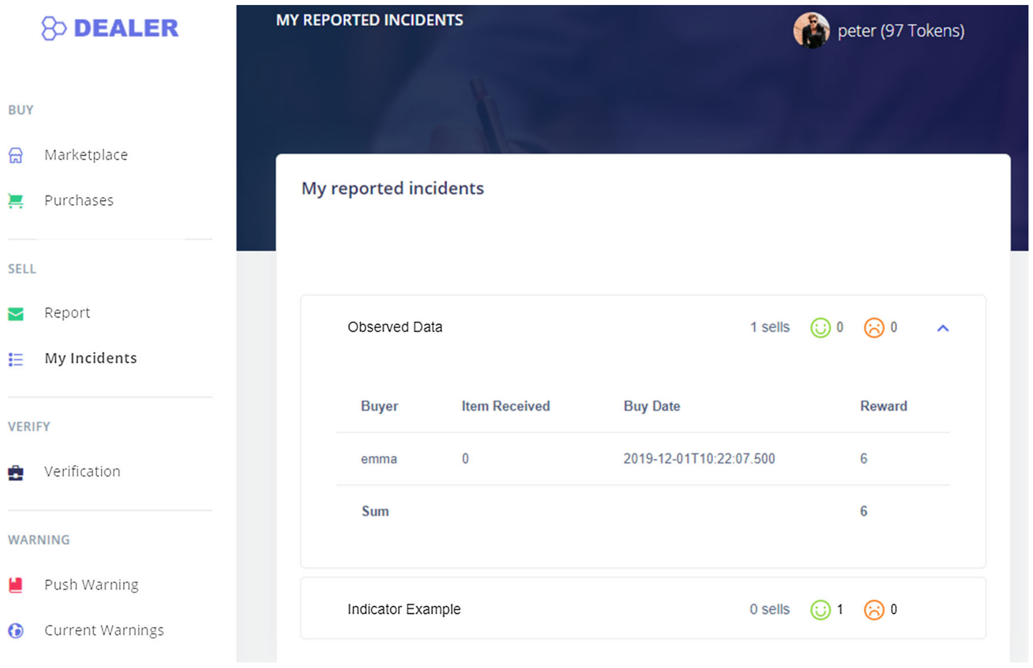


Fig. 6 Prototypical implementation of the DEALER platform



My reported incidents

Observed Data 1 sells 0 0

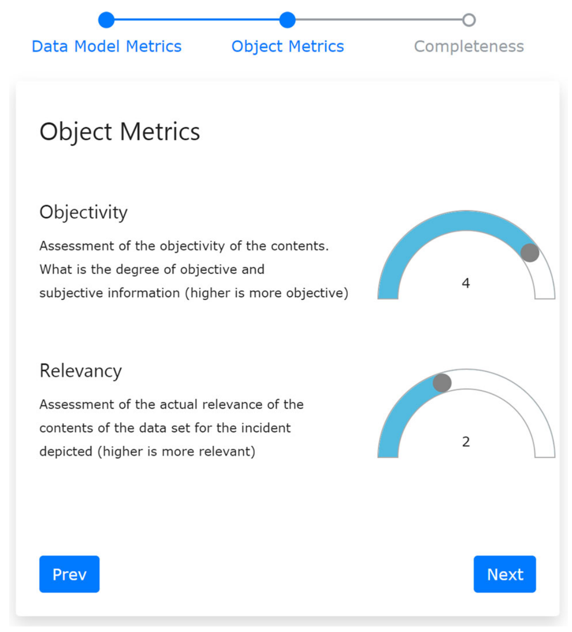
Buyer	Item Received	Buy Date	Reward
emma	0	2019-12-01T10:22:07.500	6
Sum			6

Indicator Example 0 sells 1 0

Fig. 7 User interface of the DEALER platform

as its current verification status and price. Buyers can also manage past purchases and re-download previously bought information at any time. The area *SELL*, allows sellers to report an incident to the blockchain. Such a report can contain a title, a short description, the corresponding industry sector, the actual incident data and a sale price. Incidents are encrypted using AES-256-CBC before being uploaded to IPFS. After the DHT upload, the hash reference and metadata are submitted to the smart contract. If the incident was intended for sale, RSA-encrypted copies of the AES symmetric keys are shared with the verifiers using their public keys stored on the blockchain. Besides reporting, sellers can manage past reports and view the verification status and number of their successful sales.

The *VERIFY* section allows the user to act as a verifier for an incident. The verifier is presented with a list of all incidents assigned for verification. For each individual incident, the verifier is presented with a wizard as shown in Fig. 8. The wizard sequentially requests input for the quality criteria defined in Sect. 4.3. The verification results are arithmeti-



Object Metrics

Objectivity

Assessment of the objectivity of the contents. What is the degree of objective and subjective information (higher is more objective)

4

Relevancy

Assessment of the actual relevance of the contents of the data set for the incident depicted (higher is more relevant)

2

Fig. 8 DEALER verification wizard

cally averaged after submission and sent to the platform in a blockchain transaction. Although the prototypical application allows a weighting of the individual quality criteria, this was not implemented within the demonstration prototype for reasons of clarity.

Finally, the area *WARNING* allows authorities to issue warnings on current threats to platform participants. Warnings contain informational text and structured incident information for particularly dangerous threats.

The source code for the prototype can be downloaded at the project repository.[10] A live version of the DApp is available online,[11] and the deployed EOS contract can be inspected on the EOS Kylin testnet.[12]

5.4 Implementation challenges

EOS. EOS developer tools posed some challenges, as the development environment EOS Studio crashed frequently during our tests. Some features did not work as advertised or did not work at all. Another sticking point is that debugging is not possible within the environment and even console outputs are only accessible in a cumbersome way. However, many of these issues were improved with subsequent updates during our research.

[10] https://github.com/Dealer-Platform/.

[11] https://dingfest.ur.de/dealer/.

[12] https://kylin.bloks.io/account/eosdealeradm.

Furthermore, achieving scalability of the smart contract is not trivial. EOS allows a maximum of 150ms CPU time per transaction, so performance must be kept in mind while developing the smart contract. For example, loops over table entries must be avoided, since they will lead to exceeded transaction CPU time as tables grow larger. Instead, indexes should be added on the required columns using the `multi_index` table feature. Additionally, page load times increased with an increasing number of incidents. This issue can be resolved by setting appropriate limits on `get_table_rows` queries to the EOS node and paginating results.

IPFS. IPFS is based on a content-addressed DHT data structure. This means that the address of data changes when the data are mutated by an update. It should be kept in mind that the DEALER platform needs to provide a single address for buyers and verifiers to retrieve decryption keys from a seller. With IPNS, IPFS provides a way to get a single fixed address, whose link target (i.e. a folder with keys) can be updated dynamically. Unfortunately, this address is tied to the IPFS node, which means that each user has to operate their own IPFS node. While this may be seen as a limitation of our DEALER implementation, it also comes with the advantage of user data sovereignty. Even if other IPFS nodes go offline, data will remain stored locally once it has been retrieved from the IPFS DHT.

6 Evaluation

After presenting the prototype design, we now evaluate whether the chosen blockchain platform fits the needs of threat intelligence reporting. Since EOS supports > 1000 transactions per second [29], we do not expect throughput to become a bottleneck. However, there are costs associated with transacting on a public blockchain, which we evaluate in Sect. 6.1. Additionally, we consider computation times, network latency and storage requirements in Sect. 6.2.

6.1 Transaction costs

Smart contracts on EOS require CPU, NET and RAM to execute. CPU and NET represent the processing and network utilization of transactions and are acquired by staking EOS for a fixed time. RAM is needed to store data in the smart contract state and is purchased at a fixed price. To calculate the required stake per user to run the contract sustainably, we evaluate the resources consumed by our smart contract in Table 2. Transactions were run multiple times with differing parameters on the EOS Kylin testnet. For CPU/NET, the values represent locked currency, i.e. to share one incident per day, EOS worth 0.20€ must be staked permanently. For RAM, the costs cumulate with each executed action and are

Table 2 Resources consumed by the EOS smart contract.

Action	CPU (stake)	NET (stake)	RAM (purchase)
Sharing	1.76 ms, 0.201€	0.256 kb, 0.0005€	0.755 kb, 0.088€
Verification	0.58 ms, 0.067€	0.120 kb, 0.0002€	0.000 kb, 0.000€
Purchase	1.07 ms, 0.065€	0.112 kb, 0.0002€	0.153 kb, 0.018€
Warning	0.74 ms, 0.084€	1.71 kb, 0.0034€	1.896 kb, 0.221€

EOS price: 2.00€, RAM price: 0.058 EOS/kb, CPU cost: 0.05 EOS/ms, NET cost: 0.001 EOS/kb

thus much higher. For this reason we now focus on RAM costs.

In the following we estimate the costs of the platform based on a real-world example. Therefore, we assume that the platform will be used for the reporting obligations of critical infrastructures in Germany. According to the Federal Office for Information Security (BSI), it is estimated that around 250 reports are carried out annually in 9 industry sectors [31]. The EOS RAM needed to store 250 incidents costs 22€ per year at the current conversion rate. The verifications do not cost any RAM since they only modify storage entries and don't add data.

We assume that participating companies are particularly interested in information from their sector (on average 28 reports per sector). According to the BSI, 1648 institutions in Germany are currently affected by the reporting obligation [31]. We thus estimate about 1648 * 28 = 46,144 purchases to be made in ongoing operations (823€). Additionally, we assume that authorities may issue warnings about once a month (3€). In summary, we expect a total RAM cost of 848€ to store all platform interactions occurring in one year. This is quite a feasible amount, considering that it covers more than a thousand institutions.

6.2 Performance

We evaluate the performance of our prototype with regard to user request latency (network latency and computation time) as well as storage requirements.

Request latency. We evaluate the performance of the server component locally on a machine with an i7-8550U CPU and 16GB RAM, running node.JS v10.17. The ping latency from the local machine to the go-ipfs v0.6.0 node running on a Raspberry Pi 3B is $\mu = 0.9\,\text{ms}, \sigma = 1.3\,\text{ms}$, while the ping latency to the EOS Kylin network node is $\mu = 12.0\,\text{ms}, \sigma = 0.9\,\text{ms}$ (100 pings). Request latency consists of transmission latency and server-side computation time. We measure the full request latency by timing `curl` requests with a bash script. Computation time is measured by tracking execution time of routes within the node.JS express instance for these requests. Therefore, transmission latency and client rendering speed are not included in measurements. However, these delays are negligible after the initial download of JS, CSS and image assets.

Table 3 Request Latency (RL) and Computation Time (CT) in ms

Action	RL μ	RL σ	CT μ	CT σ
W - Sharing	2185	327	2112	328
W - Verification	2453	137	2379	136
W - Purchase	2432	204	2355	202
W - Warning	1813	287	1738	286
R - Marketplace	1106	138	1041	138
R - Purchases	924	118	856	117
R - Report	65	6	1	1
R - My Incidents	1534	175	1464	168
R - Verification	1059	128	992	128
R - Dispute	930	132	863	132
R - Push Warning	404	34	338	34
R - Current Warnings	66	2	1	1
R - User Profiles	1050	95	983	95

As for the testing setup, there are 211 existing incidents in the smart contract, 108 of which are assigned to our test user for verification. We test each operation 100 times, including item upload, verification and purchase operations. We executed the tests in the order shown in Table 3.

The results show reasonable latencies of 1–2 s. Generally, the loading time for POST requests is higher, since the server first parses the request and then also prepares the webpage for the returned page. For example, to obtain the effective computation time of *W - Sharing*, the latency of *R - My Incidents* must be subtracted. In practice, the GET request latency can be removed by offering a POST-only endpoint for automated reporting.

In summary, the latencies should be appropriate for normal usage. Additional front-end optimizations such as pagination can further optimize page load times once a large number of incidents is stored on the platform.

Storage requirements. Storage needs of the EOS platform are covered in Sect. 6.1, so we now focus on off-chain storage of incident data and file keys on IPFS. We uploaded a small fixed-size (38 bytes) incident *m* times. During our initial experiments we found that storage consumption grew exponentially, but was significantly reduced after running IPFS garbage collection. Garbage collection deletes local copies for old versions of data no longer in use, for example for updated file key entries. Therefore, we run the garbage collector before taking each measurement. This ensures that storage consumption is measured correctly and does not

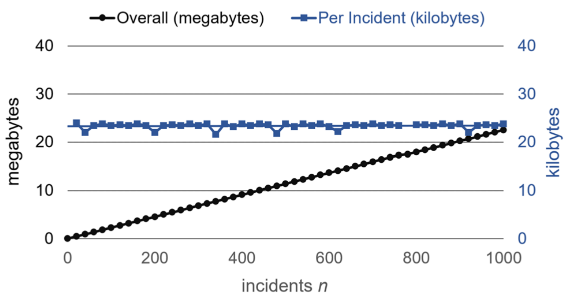


Fig. 9 IPFS storage consumption with increasing number of uploaded incidents

include duplicate entries from prior versions of the user's shared IPNS folder.

Figure 9 shows storage consumption of a single IPFS node with an increasing number of uploaded incidents. Storage consumption increases linearly with each uploaded incident. The total overhead is about 22 KB for each additional uploaded incident. 8 KB are added through preprocessing, which consists of the AES encryption of the incident, storing the ciphertext as base64 encoded string, and uploading file keys for the verifiers. The remaining 14 KB are due to IPFS internal data organization and tracking. We also tested a larger 1,032 KB incident and observed 380 KB total overhead. 340 KB are due to preprocessing, and another 40 KB are added by IPFS. The amount of overhead increases linearly with larger files, since the overhead originates largely from encryption and encoding (i.e. 2 MB incident $\sim$ 2.76 MB ciphertext).

6.3 Expert interviews

In addition to the evaluation of the transaction costs and the performance of the platform, we conducted several expert interviews to demonstrate the overall validity of our approach. The goal was to show that the intended implementation of the exchange platform offers real benefits for the industry. In this context, two specific questions were addressed. On the one hand, it was investigated to what extent the planned incentive system offers actual stimuli for companies to use it. On the other hand, it was investigated to what extent the integrity assurance measures can offer added value for companies in the reporting process. In addition to questions regarding the efficiency of the built-in incentive system, a further goal of the interviews was to get an impression of the usability of the platform as a whole in order to explore possible optimization opportunities. Accordingly, the interviews also covered the exchange process, the usability of the user interface and perceived security of the platform.

The interviewees are four security experts from different industry sectors. We conducted interviews with a Project Manager of a SME operating in the area of secure cloud services, with a security expert of a large corporate data center, with an academic researcher in the Field of Cyber Threat Intelligence as well as with a Security Consultant of a SME operating in the field of security consulting. We have designed the selection in such a way that all persons interviewed have extensive knowledge in the field of IT security and can therefore adequately assess the security benefits of the platform for their companies.

The expert interviews were designed according to the semi-structured approach of Lazar et al. [32] and are subdivided into the following 5 phases.

Phase 1—Introduction. At the beginning of the interviews, each interviewee was first asked about his or her knowledge as well as the extent of experience in the field of IT security and their knowledge of currently existing reporting obligations. The participants were also asked about their current position in the company and their budget responsibility in the area of IT security. The participants were also encouraged to indicate problems with the interview process at an early stage.

Phase 2—Incentive structure. The objective of the first thematic interview phase was to examine the benefits of financial incentives for the exchange process. To this end, the DEALER platform was first presented to the participants and the underlying idea was explained in detail. Subsequently, the participants were asked whether such an incentive system would be suitable for may be of interest to companies in principle. In this context, the participants were also asked what basic conditions would have to be fulfilled for their active participation. Finally, we asked if the participants can think of ways to abuse the system, or if they had concerns that they could be cheated by other participants.

Phase 3—Integrity features. The goal of the interview's second thematic phase was to assess the usefulness of the platform's integrity assurance and non-repudiation mechanisms. In order to achieve this, the participants were asked whether they had already been confronted with reporting obligations and whether their company is subject to reporting requirements. Subsequently, the participants were asked whether they saw a concrete benefit in the provision of integrity assurance and non-repudiation mechanisms and how this would be useful for them.

Phase 4—Platform usability. After evaluating the basic benefits of the concept in the previous interview phases, this phase deals with the actual implementation of the platform. The goal was to evaluate the usability and the benefit of the user interface as well as the exchange and reporting process. In order to obtain meaningful results, the participants were given access to the platform and only a brief explanation of the basic features of the platform was given. The participants were then given two tasks. First, they had to post a fictional security incident for sale on the platform and at the same time report it to an authority. In the second step, the participants were then to get an overview of the market situation and buy information about a security incident. In this phase we pay special attention to how well the participants understand the platform and how they handle it. In addition to mere observation, the participants are also asked about their experience using the platform.

Phase 5—Wrap-up. In this last phase of the interview, a summarizing discussion is conducted. Finally, the participants are asked again about their overall impression of the platform and whether they could imagine using such a concept in an operational context. In addition, the participants are asked about further points of criticism and possible suggestions for improvement.

6.4 Interview results

The interviews lasted between 30 and 80 min. Longer interviews were mainly due to extensive discussions with the participants about the platform and possible application scenarios of the approach. At the same time also the large interest of the participants in the presented beginning showed up. All in all, the interviews led to a whole range of additional insights regarding the incentive structure, integrity features and platform usability.

Incentive structure. In the first part of the interview, the participants were asked whether the proposed incentives were interesting, whether participation in the platform was conceivable for them and whether they had any concerns about using it. Generally, the paid exchange of incident information was met with great interest. However, it also became clear that the platform would essentially be used for the exchange of non-critical incidents. In this section of the interview, most of the interviewees placed a very high value on automation and low personnel costs. Specifically, platform participation was considered attractive if the platform would save time and personnel expenses. From the interviewees' point of view, this can be achieved especially by providing high-quality reports, as this can save a lot of time in the evaluation and use of information. It also became clear that for companies, the verifiers and quality assurance play the central role on the platform. To make quality assurance transparent, interviewees suggested introducing certification for the verifiers, which could, for example, be performed by authorities. An essential participation prerequisite was the availability of an API for automated incident processing, in order to increase efficiency and avoid expensive manual labor. Another central factor for the use of the platform is the legal security of its use. On the one hand, it was pointed out that incident reporting can only be carried out if legal certainty is established. Another criterion was the possible use of SLAs and general terms and conditions.

Integrity features. In the second part of the interview, the participants were asked about the mechanisms of integrity assurance and non-repudiation on the platform. Overall, the interviewees see a significant value benefit from these functionalities, which is particularly evident in the context of reporting obligations or insurance-related claims. They see clear potential for automation and reduction of bureaucracy. Especially the possibility to report on time, based on facts, irrevocably verifiable and tamper-proof were considered important features by the interviewees. It was emphasized that this feature is particularly interesting in cases where very high penalties are imposed for failure to report. However, the interviewees also pointed out various pitfalls and problems in implementing these features. It was shown that integrity assurance could also be carried out by public authorities and that for a real world implementation, various funded projects involving the authorities concerned would certainly be necessary.

Platform usability. Finally, the participants were asked about usability aspects of the platform. Overall, it can be stated that all participants understood the platform in principle and were able to use it completely after a short time. The interviews also consistently provided positive feedback on the proof of concept presented. The verifier user interface was particularly positively highlighted. At the same time, many suggestions for improvement were also made, especially with regard to productive use of the application. For example, it was suggested to integrate various additional information on legal implications of actions on the platform as well as the possibility to provide data sets with SLAs or terms and conditions. Furthermore, it was pointed out that in production use, extensive tools for the presentation of data set metadata are necessary in order to make clear and efficient purchase decisions. In this context, it was also suggested to introduce a subscription function for relevant sellers and to provide API access to increase the efficiency of the platform.

7 Discussion

In this section we discuss the results of this work. For this purpose, the previously defined requirements are reviewed in Sect. 7.1 and compared to the actual results achieved in the prototype. Subsequently, we discuss security concerns for the platform in Sect. 7.3.

7.1 Requirements

Reporting requirements. At the beginning of this work, Sect. 3 defined various requirements for a platform that simultaneously complies with legal requirements and offers incentives for the exchange of CTI information. Specifically, we defined *integrity* and *availability* of data as well as the *non-repudiation* of reports as target values for compliance with legal requirements. The decentralized blockchain technology used provides the necessary basic conditions to build a platform that is compliant with these requirements. One of the most important features of a blockchain is the assurance of data integrity using the decentralized ledger technology. Our solution assures *integrity* by including a hash of the data on-chain. Due to the EOS blockchain's immutability, this hash can be traced back to the original upload transaction and authenticated with the sender's signature.

Availability. The presented concept has, as previously stated, increased demands on the availability of the platform. In general, blockchains also offer a very high availability of the network nodes as pointed out by Weber et al. [2]. This results from the fact that the blockchain nodes are geographically distributed and run in a highly redundant manner. At the same time, one of the main restrictions of blockchain systems is that write-access is often limited, which may result in availability drawbacks. This is mainly the result of the low number of possible transactions per second of the considered blockchains Ethereum and Bitcoin. Since the EOS blockchain exceeds the possible transactions per second of these networks by orders of magnitude [29], restrictions of write availability are unlikely. It should also be emphasized that the EOS network is distributed over the entire globe,[13] which makes the availability of the network relatively independent of local events. This problem can be tackled in various ways with the EOS chain. On the one hand, it is possible to increase the available resources for the current project by increasing the share contributed. If you have even higher availability requirements, the block chain can also be set up with your own block producers. An example for such a split with own block producers is the Ultra/UOS[14] project. In this example, an own EOS blockchain was created to meet the high demands on throughput and availability within online games.

In the presented prototype, we store metadata of each reported security incident on the EOS blockchain in a publicly accessible manner. In order to establish a reference for *non-repudiation*, a timestamp is included in the incident metadata proving the report's existence. A reference to the reporting EOS wallet is included to link the report to the reporter's EOS wallet. The full incident data are stored on the IPFS DHT and replicated by the incident's seller and verifiers, ensuring *availability* of off-chain data through sufficient redundancy.

In addition to this, the prototype also provides the necessary tools to protect personal data within reports according to legislations such as the GDPR. To achieve this, the exchanged information is processed in an encrypted form on the platform. Each data flow is addressed to an explicit recipient and protected with the corresponding public key. This ensures that only the receiving authority can view the reported information. In the case of an exchange on the marketplace, the data are also encrypted and assigned to a buyer and verifier as specific recipients. However, since the data are transferred to different recipients, the mere assignment to the recipient is not sufficient for information and privacy protection. According to this, the offering company must decide here which data may be passed on to recipients. Both the interests of the company and the legal situation must be taken into account.

Incentives. As shown above, incentives represent are a necessary condition for an active exchange between the parties involved. In order to be able to implement such incentive procedures, we created marketplace within the platform for the mutual exchange of CTI information. Participants can offer their incident information at the marketplace in return for payment. This gives them the ability to compensate costs incurred in the detection and recording process and thus provides a financial incentive to participate in the platform. Another focus of the platform is to ensure sustainability of the implemented incentive structure. Verifiers ensure the data quality of the traded CTI information as well as functions that guarantee transactional fairness for both buyer and seller. Verifiers and sellers have an incentive to host incident data on IPFS since they profit from incident sales.

7.2 Comparison to other platforms

Overall, it can be concluded that the platform for the exchange of CTI information presented in this work offers several specific advantages over existing CTI sharing platforms. Traditional systems usually rely on trust in a Trusted Third Party (TTP) to implement the data protection goals. In contrast to this, the decentralized DEALER system guarantees these protection goals without the need for a specific trust relationship. The availability of the platform is distributed among different independent actors and no central actor is required for integrity proofs. Moreover, the implemented marketplace for the exchange of information is likewise not dependent on the trustworthiness of actors. Within the implemented smart contract, the sales process as well as the selection of verifiers is predefined and transparent for all participants.

7.3 Security

Free-riding verifiers. An important consideration is prevention of free-riding verifiers. Every verifier periodically receives free access to a randomly selected incident. As a result, verifiers must be punished if they do not perform verification as requested. If a verifier repeatedly fails to verify

[13] https://glass.cypherglass.com/map/main/top50.

[14] https://www.ultra.io.

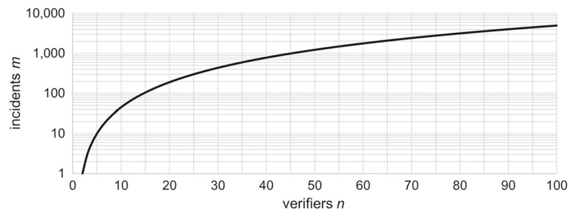


Fig. 10 A pair of verifiers is assigned to the same incident every m incidents, given n verifiers (y-axis is log scale)

assigned incidents in active status, other verifiers may start a multisignature vote for verifier removal. This encourages verifiers to only remain active when they intend to verify, to avoid losing their verifier status.

Content reselling. Reselling information is a common concern for data marketplaces [18]. As with all digital marketplaces, content reselling *outside the platform* cannot be fully prevented. Both buyers and verifiers may attempt to resell incidents they obtained through DEALER. In practice, this is discouraged by the difficulty of selling digital goods without trusted intermediaries [22]. The smart contract of the DEALER platform replaces intermediaries and provides certainty for buyers that they will receive the incident. Therefore, it is more difficult for illegal re-sellers to find buyers outside the platform without the market-making aspects of DEALER. To prevent reselling content *on DEALER itself*, the verifier system is in place to prevent it. The hash of the shared incident data is stored in the smart contract, allowing the identity of the original author to be clearly established through the timestamp and the signing public key of the transaction. Uploading duplicate incidents with the same hash is prevented by the smart contract, but resellers can slightly modify the incident to change its hash. Still, in the long run the similarity checks introduced in Sect. 4.3 reveal duplicates. If a duplicate is recognized, verifiers may submit a low rating. Similarly, buyers are likely to notice that they received a duplicate and rate the incident poorly, leading to a decreasing rating. This discourages potential buyers and lead to decreasing profits from reselling attempts.

Sybil attacks. Sybil attacks involve attackers being able to create new identities cheaply to manipulate the application. They can be mitigated by introducing nontrivial barriers to entry. On the DEALER platform, this threat mainly applies to sellers and verifiers. Sybil sellers could flood the platform with incidents to overwhelm verifiers. Sybil verifiers could dilute the quality assurance verifiers are supposed to provide. Therefore, as established in Sect. 4.1, both sellers and verifiers need to deposit cryptocurrency to create an account. Verifiers additionally need to prove their physical identity on registration. These measures present a significant obstacle for creating Sybil users.

Verifier collusion. The platform requires a minimum number of verifiers to ensure their assignment is sufficiently random to deter collusion. If assignment is not random, sellers may collude with verifiers to ensure incident verification. Alternatively, a pair of verifiers may collude during dispute resolution. The binomial coefficient determines the probability of assigning two verifiers to the same incident (n is the number of verifiers, and $k = 2$). As shown in Fig. 10, with 15 verifiers the probability is $< 1\%$, while with 50 verifiers it is $< 0.1\%$. Hereby we determine 15 verifiers as a safe minimum number of verifiers to safely operate the platform. Since verifiers may be temporarily inactive, a higher number is preferable in practice. With an expected amount of 250 reports annually (Sect. 6), each pair of verifiers shares only 2–3 incidents per year, which provides little incentive for collusion.

Even if an attacker is able to guess the pseudorandom number, the potential impact of such an attack is low. The background for this is the corresponding attacker model. At best, the attacker could assume the seller role and choose which verifiers are assigned to an uploaded incident. If these verifiers are controlled by the attacker, he may generate false ratings. By making fake incidents seem attractive, this could trick potential buyers into purchasing the fake incident. However, this would quickly become apparent, since buyers would rate such incidents low. If buyer ratings significantly diverge from verifier ratings, such incidents can be marked as potentially fraudulent in the DApp. Colluding sellers and verifiers are also registered by name on the platform and can be banned through majority consensus (Sect. 4.1).

Incident confidentiality. A compromise of the RSA or AES encryption scheme might compromise the confidentiality of the incidents stored on IPFS. Since IPFS data are stored on publicly available nodes, confidentiality is an inherent problem that can only be counteracted by encryption. This is especially the case as it is not possible to prevent an attacker from downloading the entire history for later decryption. However, we consider this scenario to be less problematic for various reasons. On the one hand, the procedures are state-of-the-art encryption technology and it can be assumed that they will be considered secure for many years to come, while the benefit of decrypted information on security incidents will decrease significantly over time. On the other hand, it can be assumed that the participants of the platform do not trade highly confidential data via the platform, since it is known that at least the validators must be given insight into the data and a large part of the data are available for sale on the platform anyway. Accordingly, the confidentiality of the data essentially relates to the protection of participation incentives. A possible compromise of the encryption schemes can additionally be counteracted by re-encrypting the data with a secure procedure, at least partially. If it is possible to decrypt incidents without purchase, participation for sellers and verifiers would be eliminated. Accordingly, such a change to newer procedures would be necessary at an early stage.

8 Conclusion

In this work we presented a fully decentralized model for sharing CTI. It is designed with legal and privacy requirements in mind and ensures sustainable sharing using cryptocurrency-based incentives. We implemented the DEALER platform based on the EOS blockchain and IPFS DHT and demonstrated its practical feasibility. On the platform, structured incident information is exchanged pseudonymously. Randomly selected verifiers use a set of objective CTI quality indicators to bootstrap incident reputation and help buyers select fitting incidents. Buyers and sellers are protected through dispute resolution mechanisms and exchange items based on cryptocurrency incentives.

Beyond our model and prototypical implementation, an integration with existing incident discovery, reporting and visualization systems is essential to the platform's practical viability. For example, the incident information currently available in plaintext could be enriched by a visualization system such as the one presented by Böhm et al. [33]. Based on such integrations, the platform can be deployed on the public EOS blockchain and tested with a larger number of users. In this scenario, price discovery mechanisms and their relationship to incident data quality can be analyzed. While our infrastructure is developed with privacy in mind, future work should ensure privacy and compliance with legal requirements (i.e. GDPR) in practice.

Acknowledgements We would like to thank our reviewers for their in-depth feedback and comments, which contributed significantly towards the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was supported by the Federal Ministry of Education and Research, Germany, as part of the BMBF DINGfest Project (Grant No. 16KIS0501K) (https://dingfest.ur.de).

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Kannengießer, N., Lins, S., Dehling, T., Sunyaev, A.: What does not fit can be made to fit! trade-offs in distributed ledger technology designs. In: Bui, T. (ed.) 52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8–11, 2019, pp. 1–10, ScholarSpace (2019). http://hdl.handle.net/10125/60143
2. Weber, I., Gramoli, V., Ponomarev, A., Staples, M., Holz, R., Tran, A.B., Rimba, P.: On availability for blockchain-based systems. In: 36th IEEE Symposium on Reliable Distributed Systems, SRDS 2017, Hong Kong, Hong Kong, September 26–29, 2017, pp. 64–73 (2017). IEEE Computer Society. https://doi.org/10.1109/SRDS.2017.15
3. Schwartz, A., Shah, S.C., MacKenzie, M.H., Thomas, S., Potashnik, T.S., Law, B.: Automatic threat sharing: how companies can best ensure liability protection when sharing cyber threat information with other companies or organizations. Univ. Mich. J. Law Reform **50**, 887 (2016)
4. Laube, S., Böhme, R.: Mandatory security information sharing with authorities: implications on investments in internal controls. In: Ray, I., Sander, T., Yung, M. (eds.) Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security, WISCS 2015, Denver, Colorado, USA, October 12, 2015, ACM, pp. 31–42 (2015). https://doi.org/10.1145/2808128.2808132
5. Bauer, S., Fischer, D., Sauerwein, C., Latzel, S., Stelzer, D., Breu, R.: Towards an evaluation framework for threat intelligence sharing platforms. In: 53rd Hawaii International Conference on System Sciences, HICSS 2020, Maui, Hawaii, USA, January 7–10, 2020, pp. 1–10, ScholarSpace (2020). http://hdl.handle.net/10125/63978
6. IBM Corporation: X-Force Exchange. https://exchange.xforce.ibmcloud.com/
7. Facebook Corporation: Facebook Threat Exchange (2019). https://developers.facebook.com/programs/threatexchange/
8. Wagner, C., Dulaunoy, A., Iklody, A.: MISP—the design and implementation of a collaborative threat intelligence sharing platform. In: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pp. 49–56 (2016)
9. Luatics: OPENCTI. https://www.opencti.io/en/
10. Liu, C.Z., Zafar, H., Au, Y.A.: Rethinking FS-ISAC: an IT security information sharing network model for the financial services sector. CAIS **34**, 2 (2014)
11. Wagner, T.D., Mahbub, K., Palomar, E., Abdallah, A.E.: Cyber threat intelligence sharing: survey and research directions. Comput. Secur. **87**, 101589 (2019). https://doi.org/10.1016/j.cose.2019.101589
12. Serrano, O., Dandurand, L., Brown, S.: On the design of a cyber security data sharing system. In: Proceedings of the 2014 ACM Workshop on Information Sharing 38; Collaborative Security. ACM, New York, USA (2014), WISCS '14, pp. 61–69
13. Dandurand, L., Kaplan, A., Kácha, P., Kadobayashi, Y., Kompanek, A., Lima, T.: Standards and tools for exchange and processing of actionable information. November (2014)
14. Brown, S., Gommers, J., Serrano, O.: From cyber security information sharing to threat management. In: Proceedings of the 2nd ACM

Workshop on Information Sharing and Collaborative Security, pp. 43–49 (2015)
15. Mohaisen, A., Al-Ibrahim, O., Kamhoua, C., Kwiat, K., Njilla, L.: Rethinking information sharing for threat intelligence. In: HotWeb 2017—Proceedings of the 5th ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies (2017)
16. Sauerwein, C., Sillaber, C., Mussmann, A., Breu, R.: Threat Intelligence Sharing Platforms : An Exploratory Study of Software Vendors and Research Perspectives, 13. Internationale Tagung Wirtschaftsinformatik, WI 2017, St. Gallen (2017)
17. Sillaber, C., Sauerwein, C., Mussmann, A., Breu, R.: Data quality challenges and future research directions in threat intelligence sharing practice. In: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pp. 65–70 (2016)
18. Alexopoulos, N., Vasilomanolakis, E., Roux, S.L., Rowe, S., Mühlhäuser, M.: TRIDEnT: Building Decentralized Incentives for Collaborative Security (2019). arxiv:1905.03571
19. Gong, S., Lee, C.: Blocis: blockchain-based cyber threat intelligence sharing framework for sybil-resistance. Electronics **9**, 521 (2020)
20. Homan, D., Shiel, I., Thorpe, C.: A new network model for cyber threat intelligence sharing using blockchain technology. In: 10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019, Canary Islands, Spain, June 24–26, 2019, pp. 1–6. IEEE (2019). https://doi.org/10.1109/NTMS.2019.8763853
21. Shafagh, H., Burkhalter, L., Hithnawi, A., Duquennoy, S.: Towards blockchain-based auditable storage and sharing of iot data. In: Thuraisingham, B.M., Karame, G., Stavrou, A. (eds.) Proceedings of the 9th Cloud Computing Security Workshop, CCSW@CCS 2017, Dallas, TX, USA, November 3, 2017, pp. 45–50. ACM (2017). https://doi.org/10.1145/3140649.3140656
22. Wagner, E., Völker, A., Fuhrmann, F., Matzutt, R., Wehrle, K.: Dispute resolution for smart contract-based two-party protocols. In: IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2019, Seoul, Korea (South), May 14–17, 2019, pp. 422–430. IEEE (2019). https://doi.org/10.1109/BLOC.2019.8751312
23. Bundestag, D.: Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme. Drucksache des Deutschen Bundestages **18**(31), 273 (2015)
24. Schlette, D., Böhm, F., Caselli, M., Pernul, G.: Measuring and visualizing cyber threat intelligence quality. Int. J. Inform. Secur. (2020). https://doi.org/10.1007/s10207-020-00490-y
25. Gascon, H., Grobauer, B., Schreck, T., Rist, L., Arp, D., Rieck, K.: Mining attributed graphs for threat intelligence. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (Association for Computing Machinery, New York, NY, USA, 2017), CODASPY '17, pp. 15–22 (2017). https://doi.org/10.1145/3029806.3029811
26. Ayman, A., Aziz, A., Alipour, A., Laszka, A.: Smart Contract Development in Practice: Trends, Issues, and Discussions on Stack Overflow, CoRR abs/1905.0 (2019). arxiv:1905.08833
27. Bach, L.M., Mihaljevic, B., Zagar, M.: Comparative analysis of blockchain consensus algorithms. In: 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1545–1550 (2018)
28. Zmudzinski, A.: ETH Transaction Fees Hit All-Time High Second Day in a Row (2020). https://cointelegraph.com/news/eth-transaction-fees-hit-all-time-high-second-day-in-a-row
29. Larimer, D.: EOSIO Dawn 3.0 Now Available (2018). https://medium.com/eosio/eosio-dawn-3-0-now-available-49a3b99242d7
30. Xu, X., Weber, I., Staples, M.: Architecture for Blockchain Applications. Springer, Berlin (2019)
31. Bundesamt fuer Sicherheit in der Informationstechnik. Die Lage der IT-Sicherheit (2019). https://www.bmi.bund.de/SharedDocs/downloads/DE/publikationen/themen/it-digitalpolitik/bsi-lagebericht-2019.pdf?__blob=publicationFile&v=4
32. Lazar, J., Feng, J.H., Hochheiser, H.: Research Methods in Human-Computer Interaction. Morgan Kaufmann, Burlington (2010)
33. Böhm, F., Menges, F., Pernul, G.: Graph-based visual analytics for cyber threat intelligence. Cybersecurity **1**(1), 16 (2018)

1.4 A secure and auditable logging infrastructure based on a permissioned blockchain [P4]

Status:	published
Publication:	Computers & Security
Submitted:	30 November 2018
Revised:	5 July 2019
Accepted:	27 August 2019
Citation:	Benedikt Putz, Florian Menges, and Günther Pernul. 2019. A secure and auditable logging infrastructure based on a permissioned blockchain. Computers & Security 87, (November 2019), 101602.

Journal Description: Computers & Security is the most respected technical journal in the IT security field. With its high-profile editorial board and informative regular features and columns, the journal is essential reading for IT security professionals around the world.

Computers & Security 87 (2019) 101602

Contents lists available at ScienceDirect

Computers & Security

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

A secure and auditable logging infrastructure based on a permissioned blockchain

Benedikt Putz*, Florian Menges, Günther Pernul

University of Regensburg, Universitätsstrasse 31, Regensburg D-93053, Germany

A R T I C L E I N F O

Article history:
Received 30 November 2018
Revised 5 July 2019
Accepted 27 August 2019
Available online 5 September 2019

Keywords:
Log management
Secure logging
Log auditing
Permissioned blockchain
Digital forensics

A B S T R A C T

Information systems in organizations are regularly subject to cyber attacks targeting confidential data or threatening the availability of the infrastructure. In case of a successful attack it is crucial to maintain integrity of the evidence for later use in court. Existing solutions to preserve integrity of log records remain cost-intensive or hard to implement in practice. In this work we present a new infrastructure for log integrity preservation which does not depend upon trusted third parties or specialized hardware. The system uses a blockchain to store non-repudiable proofs of existence for all generated log records. An open-source prototype of the resulting log auditing service is developed and deployed, followed by a security and performance evaluation. The infrastructure represents a novel software-based solution to the secure logging problem, which unlike existing approaches does not rely on specialized hardware, trusted third parties or modifications to the logging source.

1. Introduction

Log data is produced today by most information systems used in organizations. It provides information about regular events occurring on these systems, but may also contain indicators for malicious behavior or attacks such as denial of service attacks, malware activities and other types of attacks on an organization's infrastructure. Analysis of these logs helps prevent security breaches, or enables detection and subsequent damage control when an incident has taken place (Venter and Eloff, 2003).

In case a breach is successful, it's desirable to identify the perpetrator in a forensic investigation and bring the responsible person to court. In practice however intruders may attempt to alter or delete log entries documenting the intrusion (Schneier and Kelsey, 1999). Besides being exposed to malicious modification, log records are also often processed during analysis, for example by SIEM systems (Menges et al., 2018). To be successful in a trial, the organization must be able to provide an indisputable proof of integrity for the log evidence. This proof must guarantee that no modification occurred during processing, so that the evidence remains admissible in court.

The primary requirements for legally admissible digital evidence are *relevance* and *authenticity* (Bidgoli, 2006, p. 658ff). In order for a piece of evidence to be relevant, there should be a persistent chain of custody. Reliable and verifiable evidence generation, transmission and storage are part of this chain of custody and prerequisites for authentication of evidence in court (Casey, 2011). As a result, verifiable generation procedures constitute a key requirement for auditable logging infrastructures.

Prior research has already developed various approaches to create and protect secure logs from intruders (see Section 2). A key aspect of these works is integrity preservation of evidence using write-only or access-protected storage. Recently developed blockchain technology provides a novel way to achieve these goals. Blockchain systems are highly redundant data stores with the purpose of maintaining an append-only log of transactions. Since data is shared with other independent organizations based on distributed consensus, it is tamper-resistant. If a majority of participants are honest, availability and integrity of stored data is maintained. Of particular interest to enterprise applications are permissioned blockchains, where the set of participants is authenticated.

Based on a permissioned blockchain, we develop a secure infrastructure to ensure integrity and non-repudiation of log events without a trusted service provider. It is designed to prove the existence of a log entry at the time of generation by using integrity proofs stored in a distributed auditing layer. The blockchain network storing the proofs is maintained by a consortium of independent operators. Auditors can verify the integrity of previously submitted evidence by contacting any node in the network. Automated signing, storage and integrity proof generation for each log event provide the necessary authentication and non-repudiation.

* Corresponding author.
E-mail addresses: benedikt.putz@wiwi.uni-regensburg.de (B. Putz), florian.menges@wiwi.uni-regensburg.de (F. Menges), guenther.pernul@wiwi.uni-regensburg.de (G. Pernul).

https://doi.org/10.1016/j.cose.2019.101602
0167-4048/© 2019 Elsevier Ltd. All rights reserved.

For evaluation, we create a prototype as part of the DINGfest project (Menges et al., 2018). DINGfest aims to create an open-source SIEM infrastructure and currently consists of three main components: *data acquisition, data analysis* and *forensics & incident reporting*. This work adds a fourth *data auditing* component to ensure forensic auditability.

The paper is structured as follows: After explaining prior work and some of its shortcomings, we propose a general design for secure logging based on a permissioned blockchain. For evaluation, we then build the prototype within the DINGfest infrastructure and describe our results regarding security and performance.

2. Related work

Specialized hardware or software is required to achieve the aforementioned security goals of secure logging systems. Prior work on secure logging systems can be grouped into three categories: append-only storage systems, forward-secure evolving signatures and trusted third party (TTP) notary services (Cucurull and Puiggalí, 2016). Hardware-based write-only devices are a cost-intensive solution, especially when there is a large amount of continuously generated log data. For this reason we focus on software-based techniques hereafter.

Software-based approaches use cryptographic mechanisms to detect modifications of log files. One of the earliest works on secure logging for forensic investigations was published by Schneier and Kelsey (1999). Their proposed algorithm is used for concatenating the sequence of log events and provides forward security and verifiability. Ma and Tsudik (2009) claim that the Schneier and Kelsey scheme is vulnerable to a truncation attack, where an attacker may delete entries starting with the most recent. To eliminate these flaws, they introduce a new public-verifiable forward-secure aggregation scheme. It removes the need for an additional server and reduces the storage overhead introduced by the MAC and hash chain. While such forward-secure logging schemes are tamper-evident regarding modification by intruders, they cannot prevent deletion of the evidence. Accorsi (2009) provides an overview of existing software-based secure logging protocols and highlights this weakness. The work favorably emphasizes the author's "BBox" secure logging approach, which is claimed to be the only protocol to fulfill all security requirements for transmission and storage. It relies on trusted computing modules Accorsi (2013), which require special hardware and thus introduce additional cost. Finally, all aforementioned software-based techniques require additional logging software adjustments beyond transmission to a server, which may not be possible in all organizational scenarios.

An example for a TTP-based timestamping solution is the Keyless Signing Infrastructure operated by the Estonian security firm Guardtime (2018). It is based on generating integrity proofs without cryptographic keys by aggregating hashes from different sources (Buldas et al., 2013). Through several aggregation layers, hashes of log records are aggregated in a binary Merkle tree. The system operates in fixed time intervals and produces one aggregation tree per round. The tree root hashes are stored in a custom data structure referred to as a hash calendar. The calendar's root hash is regularly published in Estonian newspapers for public verifiability (Buldas et al., 2014). The KSI method is less prone to attacks from quantum-computing based algorithms since it only relies on hashing and uses no public-key cryptography (Buldas et al., 2014). Thanks to usage of several aggregation layers the system is also highly scalable. However, besides being closed-source and fee-based, the platform also has the disadvantage of relying on the security of Guardtime's infrastructure.

Other researchers have used blockchain to assure log integrity and auditability. For example, a recent work proposed publishing the KSI root hashes to the Bitcoin blockchain (Jämthagen and Hell, 2016). Cucurull and Puiggalí (2016) developed a secure logging approach that uses the permissionless Bitcoin network to record checkpoints of local log chains. Sutton and Samavi (2017) use a local graph database to store logs and submit integrity proof digests to Bitcoin for auditability. Since all these approaches rely on the Bitcoin blockchain, the scalability is limited by its block size and throughput. Besides the Bitcoin transaction fees, costs are also incurred through the requirement to maintain a full copy of the blockchain.

Other approaches use the permissioned blockchain framework Hyperledger Fabric.[1] Ahmad et al. (2018) focus on tracking changes made to database entries by storing change excerpts on Hyperledger Fabric. Shekhtman and Waisbard (2018) store the contents of log files directly on Hyperledger Fabric. They demonstrate the feasibility of auditable logging based on a permissioned blockchain, but it is not clear whether their approaches are scalable, as no throughput and storage scalability benchmarks are presented. However, scalability is required to manage large volumes of logging data in practical secure logging deployments.

In contrast to the works described above, our proposed approach does not rely on specialized hardware, modification of the logging source or trusted third parties. Instead it uses a combination of replicated local storage and a permissioned blockchain to ensure availability and integrity. Using a permissioned blockchain over a permissionless one comes with several advantages. Permissioned blockchain systems allow for higher throughput on the order of hundreds to thousands of transactions per second (Bano et al., 2017). Additionally, transaction costs can be avoided due to the restricted set of participants, which allows using deterministic consensus algorithms.

To summarize, we contribute to the literature by reducing cost and alleviating performance limitations of prior blockchain-based secure logging approaches. To achieve this, we rely on a high-performance and low-latency permissioned blockchain, with enhanced security provided by anchoring to a permissionless blockchain.

3. System design

Following the design science research methodology by Peffers et al. (2007), we begin by elaborating requirements and objectives for the system. Based on these requirements, we consider the available options for storing data when using a blockchain system and describe a general architecture for a blockchain-based auditable logging system. We also discuss two options for operation of the blockchain network in practice.

3.1. Requirements and preliminaries

The overall objective of a secure logging system is maintaining availability and integrity of log files. Records created by the system should also have the property of non-repudiation, meaning that records verifiably correspond to an event that occurred on a specific system. Onieva et al. (2009) define five phases of a non-repudiation service. We use four of these phases (shown in Fig. 1) to guide system design in the following chapter. In secure logging, verification occurs during dispute resolution, since tampered log records may also contain valuable information about system compromise.

A prerequisite for all phases is the definition of what actually constitutes evidence. For information system logs, any record may be possibly relevant evidence. To separate availability and integrity

[1] https://www.hyperledger.org/projects/fabric.

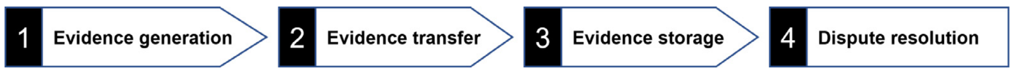


Fig. 1. Phases of a non-repudiation service, adapted from Onieva et al. (2009).

preservation, we split log records into of *evidence data* and an *integrity proof*. The evidence data consists of the actual log event and associated metadata. Specifically, it consists of log event information, a generating system identifier & signature and a timestamp. The signature is generated by the source system or the hypervisor, if the data was extracted using Virtual Machine Introspection. The integrity proof is represented by a timestamped hash of the evidence data that confirms the existence of the log event at a specified time. If this information is stored immutably, it can later be used to prove that evidence data has not been changed since creation of the proof.

Evidence generation takes place on log sources, for example systems that are connected to external networks and vulnerable to intrusion. The log files should include digital signatures signed with the private key of the generating system. In our work, a unique public/private key pair is generated for each system. Certificates issued by a public key management authority enable attribution of log events to sources.

During the evidence transfer phase, the log event is then transferred from its source to the storage system. Using an encrypted connection is imperative to maintain the chain of custody. In our solution, the transmission is subject to a number of requirements established in prior work: confidentiality, origin authentication, integrity, uniqueness and reliable delivery (Accorsi, 2009).

Evidence storage is the main focus of this work. Any log data ingested by the service should be stored in a way that preserves availability and integrity for later use. This includes preventing deletion or modification by an attacker seeking to erase traces. Confidentiality is also a concern due to potentially sensitive information contained in log records. Prior solutions described in Section 2 use specialized append-only hardware, third-party providers or local storage combined with external notary systems. Our approach achieves immutability by storing integrity proofs on a blockchain network consisting of independent nodes.

Dispute resolution occurs when an intrusion has taken place and has been recorded in log files. Both intruder and victim may attempt to deny the authenticity of the evidence. An auditor must then be able to verify that the log was not modified by anyone since generation. This evidence verification consists of signature and integrity proof verification. For signature verification, the auditor must have access to the corresponding public key certificates of the generating sources. The integrity proof is represented by a hash and a corresponding timestamp in our work and must be stored in provably unmodifiable storage. It ensures that the signed file already existed at the claimed time.

3.2. Storage design considerations

To alleviate shortcomings of prior secure logging approaches discussed in Section 2, we consider an architecture using a permissioned blockchain network. The advantage of using a permissioned network is that it does not rely on a paid third party service provider. Instead, the blockchain node operators provide the immutability service for each other. Since the operational cost is evenly shared by all members, no additional costs arise besides operating the network.

We now discuss how a blockchain-based solution could help to store log files in an integrity-preserving way. Data can be stored either on-chain or off-chain in blockchain-based solutions. On-chain storage would imply replicating the full log data across all blockchain nodes. Logging infrastructures deal with considerable data volumes, so full replication to each node would be inefficient and lead to high storage costs. An additional concern with on-chain storage is loss of privacy. Log data may inadvertently contain sensitive data like usernames or even password hashes. Since blockchain node operators are independent, they should not share potentially sensitive log data. Off-chain storage maintains data in a separate local database to uphold privacy and confidentiality requirements. Off-chain data can be linked to the corresponding on-chain transaction through its hash, provided that the data has not been modified since its hash was included in the blockchain.

To accommodate the storage limitations of blockchains, we split the log storage into on-chain and off-chain parts as suggested by Barger et al. (2018). Evidence data is stored in a local storage cluster and protected from unauthorized access to maintain confidentiality. Availability protection for the off-chain data is achieved through local replication. Commodity hardware can be used to cheaply store log data while avoiding data loss. Apache Kafka[2] is one example for a suitable publish-subscribe system that maintains a locally replicated and crash-tolerant log, while allowing other applications to interact with the data (Wang et al., 2015). To be able to detect potential corruption or modification of the off-chain data, integrity proofs are stored in transactions on the permissioned blockchain. The network is maintained by independent operators to ensure the proofs cannot be modified by any one participant. Since these operators are only semi-trusted, the network should be able to tolerate some amount of arbitrary, even malicious, behavior. These types of faults in distributed systems are also referred to as byzantine faults (Castro and Liskov, 2002). A byzantine-fault tolerant (BFT) consensus algorithm is used in our solution to tolerate up to f byzantine failures in a network of $3f + 1$ nodes.

BFT state machine replication can be implemented without a blockchain data structure to gain some throughput performance. However, secure logging requires additional authenticity and integrity guarantees as mentioned in Section 3.1. Permissioned blockchain frameworks provide these guarantees and come with other beneficial features such as audit-only nodes and APIs for external applications, so we build on them in our system architecture.

3.3. System architecture

The full architecture is shown in Fig. 2. Evidence is generated and signed by different sources like containerized applications, firewalls or intrusion detection systems. The generated evidence data is securely transferred to a log verification system. Existing protocols based on reliable syslog (IETF RFC 3195 New and Rose, 2001) fulfill the requirements for secure transmission (Accorsi, 2009) and can be used for this purpose.

Newly arriving data in the storage cluster is monitored by a separate application. For each new entry, a transaction is generated containing a hash and the current timestamp. The transaction is included in a block together with transactions arriving from other nodes in the distributed system. A new block is appended to the blockchain as soon as enough transactions are available, or when a timeout is reached. The timeout ensures that the delay between log generation and inclusion in the blockchain remains low. The

[2] https://kafka.apache.org/.

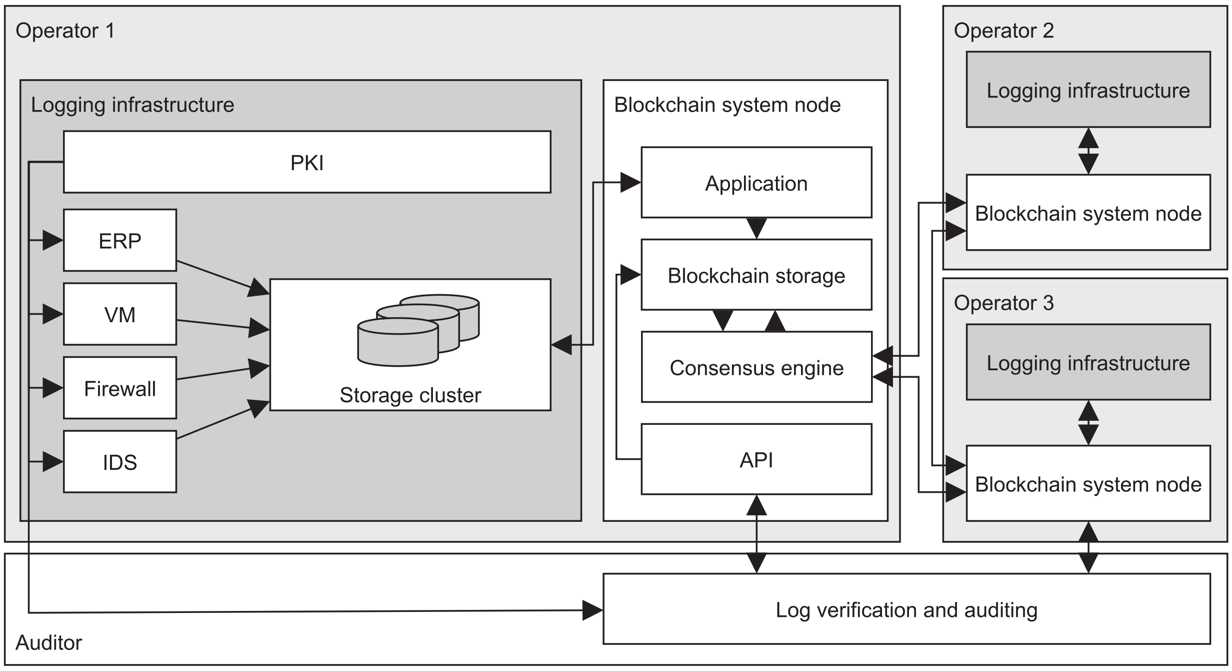


Fig. 2. Proposed design for a secure and auditable logging infrastructure.

other participating organizations also add transactions containing hashes from their own logging infrastructures.

Any auditor is able to verify evidence at a later date. The actual evidence data remains in the replicated local storage and can be provided to the auditor at request. To verify the integrity of the evidence data, the auditor first verifies its signature against valid public key certificates from the PKI. This step ensures the log file can be mapped to its source. Afterwards the data's hash is submitted to a blockchain node for verification. The server then compares the hash values stored in blockchain transactions with the hash value of the proposed evidence. If an identical hash is found, there is non-repudiable proof that identical data was submitted at an earlier time. To tolerate possibly corrupted blockchain nodes, the proof can also be requested from multiple blockchain nodes independently.

3.4. Formal logging procedure

Fig. 3 details the data flows that occur when each log entry is processed. Two processing steps **S1, S2** occur, with an optional verification step **S3**. To describe these steps, we define the following formal notation. Each organization $j \in \{1.n\}$ has an identifier ID_j, a private signing key Z_j and a corresponding public key K_j.

S1: Client application processing. We first define a log counter $k \in 1.l$ and a transaction counter $i \in 1.m$. Each newly arriving log entry L_{jk} is parsed to create a transaction payload P_{ji}, which is included in a new transaction T_{ji}. Initially, $i = k$, as one transaction is generated per log entry. *hash*() refers to a preimage-resistant hash function, while *sign*() is shorthand for a public-key signature function. It is assumed that T_{ji} is transmitted from client application to blockchain node via an encrypted channel, preventing man-in-the-middle attacks.

$$T_{ji} = (P_{ji}, H_{ji}, S_{ji}) \quad where$$

$$P_{ji} = (ID_j, K_j, L_{jk}),$$

$$H_{ji} = hash(L_{jk}),$$

$$S_{ji} = sign(Z_j, P_{ji})$$

S2: Blockchain node processing. After the transaction was propagated to the network and proposed as part of a block by the current consensus leader, the actual processing takes place on each node. A timestamp based on distributed consensus is included with each transaction and the uniqueness of the log entry is verified by each node:

$$i > 1 : H_{ji} \notin \{H_{j1}..H_{j(i-1)}\} \quad \forall j \in 1..n$$

If this condition fails, no changes to the persistent state occur as a result of the transaction. Two identical log entries with the same hash value can not be part of the system. If the condition passes and the entry is unique, a mapping of content hash to blockchain transaction hash is added to a dictionary: $(H_{ji}, hash(T_{ji}))$. This permits efficient lookups during verification. After more than two-thirds of all nodes have successfully processed the transaction, it is irreversibly committed to the ledger. We omit consensus protocol message exchanges for clarity in Fig. 3.

S3: Verification. To verify, a user may optionally submit a log entry to the client application, which requests a proof from the blockchain network by sending $hash(L_{jk})$. If the corresponding transaction is found and returned, only its signature and hash must be verified:

$$verify(K_j, P_{ji}, S_{ji}) \quad \wedge \quad K_j \in \{K_1..K_n\} \quad \wedge \quad H_{ji} \overset{!}{=} hash(L_{jk})$$

verify() is the verification function corresponding to *sign*(), returning 1 for a correct signature and 0 otherwise.

3.5. Blockchain network operation

For illustrative purposes only three nodes are shown in Fig. 2. An actual deployment in practice must consist of $n \geq 4$ nodes to be able to tolerate at least one byzantine node (see Section 5.2 for optimal node counts). Nodes should be operated by separate organizations to make it harder for both the organization and attackers to modify data on the blockchain. If a single party controls the supermajority of nodes, it could simply replace and fabricate data, forfeiting the immutability benefits of a blockchain system. The blockchain network consortium could be coordinated by industry

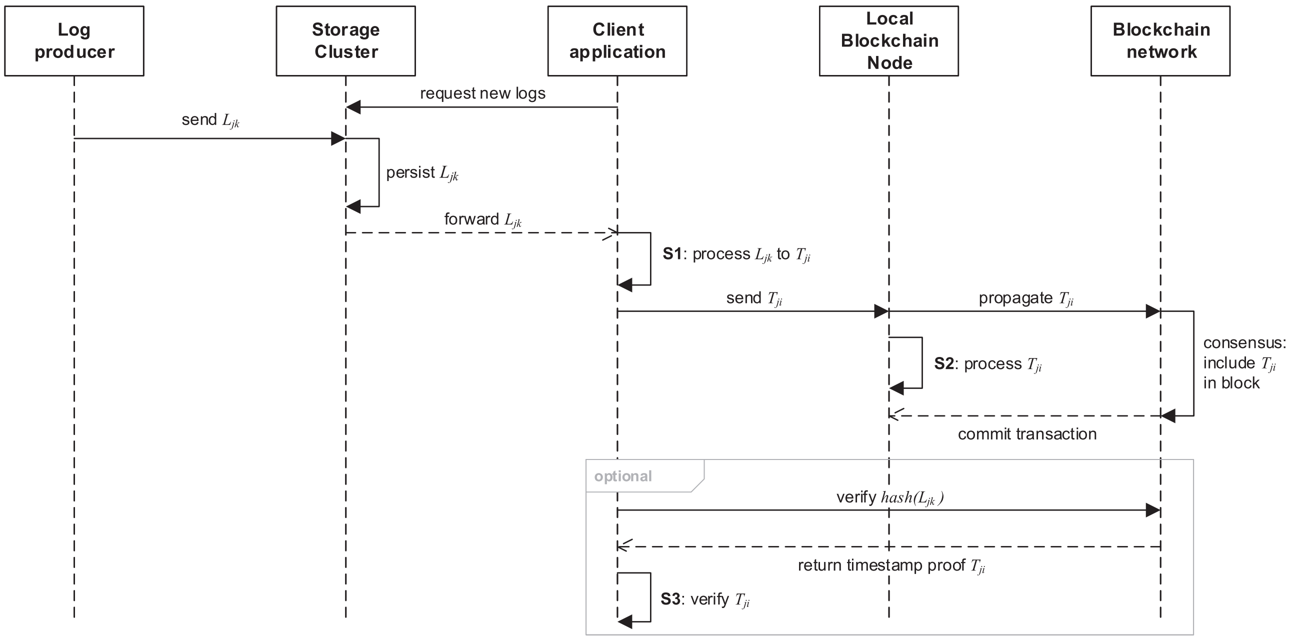


Fig. 3. Sequence diagram of log entry processing data flows.

associations. In that case independent organizations within the association would jointly maintain a network of blockchain nodes, as illustrated in Fig. 2.

Another alternative is operating the network within a single organization and spreading the nodes across multiple locations or organizational units. In that case the organization is in control of all nodes and an intruder would have to subvert nodes in multiple locations to remove traces. The organization itself could however initiate a coordinated replacement of blockchain data. This might be a concern in investigations where organizations try to hide a breach of their infrastructure. Adding an anchoring mechanism to the permissioned blockchain mitigates this possibility. Anchoring includes the latest block hash of the permissioned blockchain network in a transaction on a permissionless system like Bitcoin. These checkpoint transactions are submitted in regular intervals and cost a small fee. This allows external auditors to publicly verify the state of the private blockchain at the time of anchoring.

The advantage of this approach is that the entire logging infrastructure remains within the organization. At the same time it also incurs transaction fees similar to the permissionless blockchain approach (Cucurull and Puiggalí, 2016). Depending on the anchoring frequency this cost can add up to a substantial amount. While lowering the frequency decreases cost, it also widens the time window for possible replacement of blockchain data by the organization. To prevent this entirely, the time between checkpoints must be lower than the time an internally coordinated blockchain replacement would take.

4. Prototype

For demonstration and evaluation purposes, we create a prototype based on a SIEM reference architecture. We build on the DINGfest infrastructure created in prior work, which implements some parts of the design described above, like the storage cluster. It however currently lacks a way to ensure end-to-end integrity, auditability and non-repudiation of the original log evidence. The prototype adds this capability in the form of a blockchain-based distributed log auditing service. The service is then evaluated for security concerns and throughput/storage scalability.

The extended architecture is shown in Fig. 4. A hashing application fetches log records from the data stream and computes the SHA256 hash for each record. The input data for the hash includes the log data and its signature. The hash is then submitted to the blockchain system in a signed transaction. The receiving blockchain node validates the hash against existing hashes in the database to prevent duplicates. A timestamp is obtained in consensus with other validators and included with the transaction. Finally, a separate proof application provides a web interface, where auditors can submit evidence for validation (see also Fig. 5).

We evaluate various open source permissioned blockchain frameworks for our prototype. The evaluation criteria we deem essential for secure logging are shown in Table 1. First, we examine whether the framework currently offers a production-ready BFT consensus implementation. For performance reasons, we also want to avoid the virtualization overhead of smart contracts and thus look for frameworks offering native execution of custom logic. To protect against collusion manipulation of the permissioned blockchain, we include consensus-based anchoring to a permissionless blockchain as a criterion. To ensure a low barrier to entry, deployment effort is assessed by analyzing the number of different services and containers needed for running the framework.

We conclude that there are several advantages to using Exonum[3] over other permissioned frameworks. Firstly, other frameworks do not yet offer ready-for-use BFT consensus implementations or consensus-based anchoring. Additionally, Exonum does not use conventional smart contracts running in a virtualized execution environment. Instead, it uses natively executed *services* to implement custom logic. Similar to smart contracts, services are invoked by transactions and executed on every blockchain node, but service code must be final at compile-time and cannot be deployed dynamically at runtime. Services include the Exonum framework as a dependency and are compiled to a single binary containing both application and framework. For this reason there is no performance overhead due to virtualization. The binary can also be deployed easily without the need to maintain multiple separate containers (like in Fabric/Sawtooth/Corda).

[3] https://exonum.com.

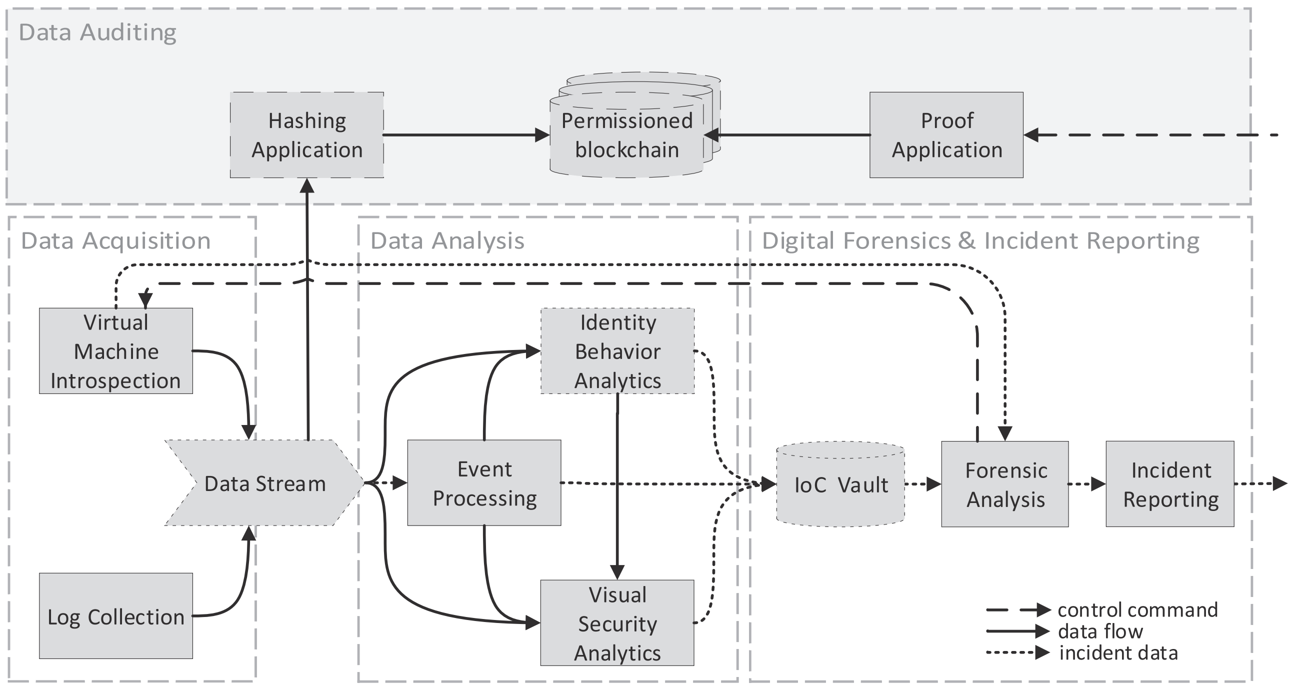


Fig. 4. The DINGfest SIEM architecture extended with an auditing layer (based on Menges et al., 2018).

Table 1
Comparison of open source blockchain framework properties.

	Fabric	Sawtooth	Corda	Ethereum	Exonum
BFT consensus	(✓)	(✓)	(✓)	(✓)	✓
Native contract execution	-	✓	-	-	✓
Consensus-based anchoring	-	-	-	(✓)	✓
Deployment effort	high	medium	high	low	low

✓ built-in (✓) custom implementation required/experimental - not available

The Exonum framework and log auditing service are implemented in Rust, a functional systems programming language focused on memory safety, concurrency and performance (Mozilla Corporation, 2018). The framework uses a byzantine fault-tolerant consensus algorithm based on PBFT (see Castro and Liskov, 2002) and supports throughput rates of up to 7000 transactions per second (BitFury Group, 2018). High throughput is important to ensure timely inclusion of each log record's integrity proof in the blockchain. The Exonum framework also provides built-in services for distributed timestamps and Bitcoin anchoring. Anchoring to the permissionless Bitcoin blockchain increases security by providing publicly verifiable checkpoints (also discussed in Section 5.1). By using a permissioned blockchain, the process of deciding on the next candidate block for anchoring is based on consensus and avoids a single points of failure (BitFury Group, 2018). Since BFT consensus is the key argument for using blockchain in our architecture, we also choose Exonum for its low overhead approach to blockchain. The features it adds apart from PBFT consensus are all useful to the proposed solution. Encrypted node connections maintain confidentiality, consensus-based distributed timestamps ensure non-repudiation of log creation time and block grouping of transactions improves consensus performance.

The prototype consists of an Exonum backend **service** and a light **client** web application, as shown in Fig. 5. The backend service runs on version 0.11 of the Exonum framework.[4]

The **frontend client** provides interfaces for blockchain inspection, monitoring and verification of log data. It retrieves data from the blockchain using the server-side backend, which redirects the read requests to the Exonum blockchain API. The server also runs a separate background application written in Rust, which continuously receives new log events from the distributed storage cluster based on Apache Kafka. The signed log data is hashed and submitted to the blockchain in a transaction. The blockchain's log auditing service verifies that the hash does not already exist in the blockchain and groups arriving transactions into blocks. New blocks are appended to the blockchain by the framework after establishing consensus with the other nodes.

The **backend service** runs on Exonum blockchain nodes and specifies the transaction data model and the available API endpoints for the light client to interact with. Our prototype reuses the timestamping service example provided by the framework, which meets all requirements of the system design and formal specification.

In practice, each network participant would deploy an instance of the blockchain node and client application on a local server. To deploy the blockchain node, participants must agree on a shared configuration file, which includes parameters like block proposal timeout and transactions per block (further described in Section 5.2). Each node's individual configuration file is then generated locally based on its private key and IP address and public key of the other nodes. The node binary, which includes both the framework and the log auditing service, is then started and ready to receive transactions. Finally, the client application must be con-

[4] https://github.com/exonum/exonum/releases/tag/v0.11.0.

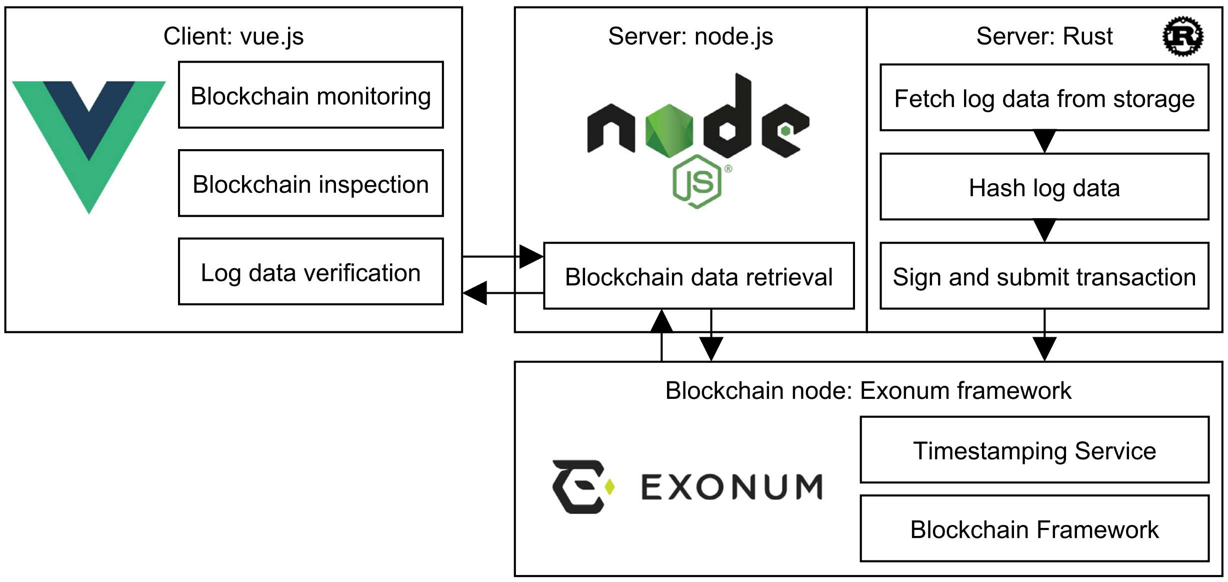


Fig. 5. Client-server architecture of the application prototype.

figured to continuously receive log records from the local storage cluster. After every participant has finished the setup phase, the system may begin operation.

5. Evaluation

Crucial aspects of the design that should be evaluated are *security* and *performance*. System security is important since vulnerabilities to attacks may question the very purpose of the infrastructure. Performance considerations are important as well to ensure scalability for larger organizations, especially since blockchain systems are known for their scalability limitations. The security evaluation is based on a structured analysis of threats, while the performance evaluation focuses on throughput and storage metrics in a cloud-based deployment of the prototype.

5.1. Security

There are four fundamental security threats to consider in a distributed system: *interception, interruption, modification* and *fabrication* (Tanenbaum and Van Steen, 2014).

Interception could lead to delays in committing timestamps to the blockchain. Intercepting a node's connection is impeded by setting up authenticated and encrypted communication channels between nodes.

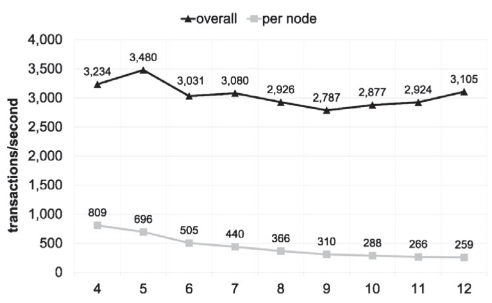
Interruption could be achieved for example with a denial of service attack. By preventing the node on the target network from communicating with other nodes, an attacker can delay outstanding log transactions. Alternatively, an intruder could attempt to gain control of blockchain nodes. Once enough nodes are compromised, it becomes possible to stall consensus or fully control the network. The exact number of nodes depends on the number of validators in the network. With a byzantine fault-tolerant (BFT) consensus algorithm, a minimum of 1/3 of all nodes is required to stall consensus and $>2/3$ to gain network control by adding/removing validator nodes. These bounds are based on the fact that more than two-thirds of validators must agree to commit a transaction in BFT consensus (Castro and Liskov, 2002). Anchoring to a permissionless blockchain would prevent this type of attack entirely. The permissioned blockchain's contents can then be verified against checkpoints published on the public blockchain to detect manipulations.

Data modification is the most significant threat to log evidence. An intruder could attempt to modify the original log record to obscure traces. As noted by Schneier and Kelsey (1999), once an attacker has control of the log source, the integrity of new logs cannot be protected. The goal of secure logging is therefore to protect log data generated prior to intrusion. Our proposed two-layer approach protects availability using the replicated storage cluster and integrity with proofs on the blockchain. To void availability of the original log records stored on the local storage cluster, the attacker would have to corrupt or delete all copies. To modify the integrity proofs on the blockchain, the attacker has to subvert or convince more than two-thirds of all nodes due to the properties of BFT consensus (Castro and Liskov, 2002). Both scenarios require significant penetration of the organization's infrastructure and are rather unlikely in practice.

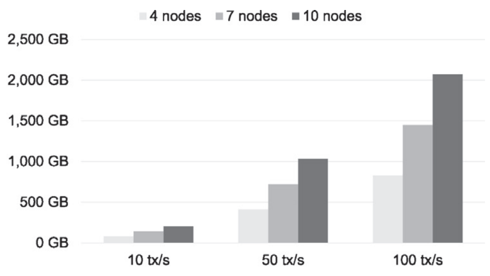
Fabrication of log entries is a concern and may invalidate authentication of the evidence. An intruder could fabricate false log entries to lead investigators astray. To that end it is necessary to gain control of the system first and compromise its private key for log generation. As recognized in Schneier and Kelsey (1999), no security measure can protect log files generated after a system has been compromised. Log files generated prior to intrusion should allow to identify the time of compromise, after which logs can no longer be considered authentic. Additionally, the organization maintaining the log infrastructure could also attempt to fabricate entries to falsely blame an adversary. This cannot be prevented, since the organization is in control of its private keys and can submit arbitrary data at any point in time signed with these keys. This possibility of organizational fabrication instead has to be excluded by legal means.

Regarding correctness of the verification, there is a non-zero probability that a fake log submitted for verification has the same hash as another valid log entry included in the blockchain. This type of hash collision can be neglected in practice however. There are 2^{256} possible hash outputs for the SHA256 algorithm used in the prototype, for which no computationally feasible collision has been found (Rasjid et al., 2017).

From a privacy perspective, the log contents remain confidential since only a hash of the data is stored on the blockchain. For preimage resistant hash functions like SHA256 it is (by current standards) impossible to find the content of the original log file.



(a) Transactions per second for varying node counts.



(b) Storage requirements for varying average throughput and node counts.

Fig. 6. Prototype performance benchmark results.

Additionally there are some operational security concerns regarding the prototype's blockchain framework. Exonum has not received any security audits, so independent validation of framework security is not yet available. Another limitation is the lack of built-in authentication and authorization for service endpoints. By default, anyone can query both private and public endpoints. This is not a critical problem for our infrastructure. If the hashing application is deployed to a separate system or container, it can be whitelisted as the only external system to use the port of the public blockchain API. Similarly, the system administrator's IP should be the only IP allowed to access the private port, since it could be used to add new peers or shut down the node. Whitelisting can be done by using the AWS security groups, iptables on Linux or a different firewall solution.

5.2. Performance

The three main concerns regarding the performance of the logging infrastructure are transaction throughput, node scalability and storage requirements. The bottleneck of the system is clearly the blockchain-based auditing layer due to its limited transaction throughput spread across all participants. Thus we set up blockchain networks of varying sizes on the DigitalOcean cloud to benchmark the prototype. We used droplet instances with 4 dedicated virtual CPU cores and 8GB of RAM. All virtual machines are set up within the same data center location. We deploy both the blockchain node and the hashing application on the same server. Consequentially, the overhead from signing and submitting transactions to the blockchain is included in the performance measurements. After some experimentation with consensus parameters we settled on a default configuration to a maximum of 10,000 transactions per block and a proposal timeout of 1 s. These parameters seemed to alleviate any bottlenecks related to system configuration. For stress testing we submit 5000 log transactions per second for 20, 30 and 40 s at a time. The transaction load is evenly distributed across all nodes. Higher loads did not increase throughput and only resulted in increased processing time after the end of transaction submission. The average observed throughput rates are shown in Fig. 6a.

The results show that overall system performance varies between 3000 and 3500 transactions per second on average, yielding between 250 and 800 processed transactions per node and second. Per-node performance continually decreases, while overall performance remains fairly stable even for a large number of nodes. The per-node performance can be interpreted as limits on the average transaction load that the system should be exposed to in each organization.

Given the per-node performance thresholds between 250 and 800 transactions per second, it is not unlikely that the log event rate exceeds system capacity. In that case a possible solution is to group log records in transactions. In step **S1** from Section 3.4, the hash H_{ji} is then computed using r concatenated log records L_{jk} as input:

$$H_{ji} = hash(L_{j[(i-1)r+1]} \mid . \mid L_{j[(i-1)r+r]})$$

The resulting load on the blockchain is reduced by a factor of $\frac{1}{r}$. However, hash verification in step **S3** changes accordingly to also require r records as input. The sequence number of the log entry may not be known to the verifier, requiring r verification requests and $2(r-1)$ additional log entries for verification (before and after the log at hand). This highlights the disadvantage to this solution: grouped records are now treated as one integrity unit. If one of the records in the unit is corrupted or no longer available, it becomes impossible to prove the integrity of all the records in the unit.

Fault tolerance maxima are achieved when the node count is equal to $3f+1$, tolerating f byzantine or failing nodes. As a result, node counts of 4/7/10 nodes can be considered efficient in Fig. 6, tolerating 1/2/3 byzantine nodes respectively.

Fig. 6 b illustrates blockchain database growth per month, based on the measured average size of 800 bytes per transaction. Since the size of the data fields is fixed, all transactions have approximately the same size. The database grows linearly with both throughput and node count. To avoid excessive storage usage, log record transactions on the blockchain may be discarded after d days, as defined by the organizational log retention policy. The ideal technical solution would be a rolling blockchain, configured to discard blocks once they are d days old. The log retention period must then be agreed upon by all node operators in advance. A rolling blockchain system has been proposed in Dennis et al. (2016), but no open source implementation is currently available. For this reason we leave this issue for future research.

5.3. Comparison with other blockchain-based secure logging approaches

To validate our results, we compare them with similar blockchain-based logging approaches (Table 2). Prior works on blockchain-based secure logging have used both permissionless (Bitcoin) and permissioned (Fabric) blockchains, but none of them have proposed an integrated approach based on anchoring. While our solution is primarily based on the permissioned blockchain Exonum, it also provides the option for consensus-based anchoring to the Bitcoin blockchain to increase tamper-resistance.

With regard to evaluation, none of the other works thus far have included benchmarks for throughput and storage constraints. Since these constraints are crucial for high frequency logging infrastructures, we have addressed them in Section 5.2. Besides con-

Table 2
Comparison with other blockchain-based approaches.

Paper	Blockchain	Permissioned	BFT	Benchmarks	Proof/data sep.	Per-entry imm.
Cucurull and Puiggalí (2016)	Bitcoin	-	n.a.	-	✓	-
Sutton and Samavi (2017)	Bitcoin	-	n.a.	-	✓	✓
Ahmad et al. (2018)	Fabric	✓	-	-	-	✓
Shekhtman and Waisbard (2018)	Fabric	✓	-	-	-	✓
Present work	Exonum	✓	✓	✓	✓	(✓)[a]

[a] Without log grouping from Section 5.2

ducting benchmarks, we also proposed avenues for increasing scalability by grouping log records or using a rolling blockchain.

Due to limited storage space in Bitcoin transactions, both Bitcoin-based approaches separate the integrity proof and the log data. The works based on Hyperledger Fabric include partial Ahmad et al. (2018) or full Shekhtman and Waisbard (2018) log data on the blockchain, leading to limited scalability. Our approach is also based on a permissioned blockchain and focuses on storing a minimal amount of data on-chain.

Per-entry immutability refers to generating one blockchain transaction per log event. Sutton and Samavi (2017) follow this approach, but use the Bitcoin blockchain, which leads to very high transaction costs and throughput limitations. These cost constraints lead Cucurull and Puiggalí (2016) to publish checkpoints instead of individual log entries. However, between checkpoints logs exist only on the local machine, which enables truncation attacks during the checkpoint interval. Mitigation is possible by reducing the checkpoint interval, but at the same time increases transaction costs. Permissioned systems do not suffer from this constraint and offer per-entry immutability, as evident from the works using Hyperledger Fabric (Ahmad et al., 2018; Shekhtman and Waisbard, 2018). Our solution combines the advantages of both solutions: it gains per-entry immutability by using a permissioned blockchain and public non-repudiation by publishing checkpoints to a permissionless blockchain, witnessed by hundreds of independent nodes.

6. Conclusion

This paper presents an infrastructure for log auditing using a permissioned blockchain to store integrity proofs. It is based on legal requirements for admissible evidence and represents an on-premise alternative to third-party solutions and specialized write-only hardware. Even without a third-party service provider, the solution achieves immutability through cooperation and data sharing between independent nodes. It permits processing of evidence for security analytics purposes while ensuring auditability of the original log record.

The security analysis shows that the system withstands attempts to intercept, interrupt or modify its processed log data. While fabrication is a concern, it cannot be completely ruled out with technical measures. Performance benchmarks show that the blockchain implementation is able to cope with very high log event frequencies of 3000 to 3500 transactions per second, depending on the number of nodes. Storage requirements are substantial however due to full replication and retention of all historical data.

In future research, an enhanced prototype could implement log rotation to reduce storage costs, as outlined in Section 5.2. While we firmly believe that Exonum currently offers the best performance for our secure logging approach, this might change in the future. In that case the prototype could also be implemented in other frameworks to see if scalability can be further improved. Practical deployments by organizations would be useful to assess adequacy of system throughput in practice. Use case studies to compare the cost structure of the proposed logging infrastructure with third-party solutions like the KSI (Guardtime, 2018) would also be valuable.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by the Federal Ministry of Education and Research (grant no. 16KIS0501K), Germany, as part of the DINGfest project (https://dingfest.ur.de).

References

Accorsi, R., 2009. Log data as digital evidence: what secure logging protocols have to offer?Proc. of the International Computer Software and Applications Conference 2, 398–403. doi:10.1109/COMPSAC.2009.166.
Accorsi, R., 2013. A secure log architecture to support remote auditing. Math. Comput. Model. 57 (7–8), 1578–1591. doi:10.1016/j.mcm.2012.06.035.
Ahmad, A., Saad, M., Bassiouni, M., Mohaisen, A., 2018. Towards blockchain-driven, secure and transparent audit logs. In: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. ACM, New York, NY, USA, pp. 443–448. doi:10.1145/3286978.3286985.
Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G., 2017. Consensus in the age of blockchains. CoRR abs/1711.0.
Barger, A., Manevich, Y., Bortnikov, V., Tock, Y., Factor, M., Malka, M., 2018. Shared Cloud Object Store, governed by permissioned blockchain. In: Proceedings of the 11th ACM International Systems and Storage Conference on - SYSTOR '18. ACM Press, New York, New York, USA. 114–114 doi: 10.1145/3211890.3211915.
Bidgoli, H., 2006. Handbook of Information Security. John Wiley, Hoboken, NJ.
BitFury Group, 2018. Exonum Documentation.
Buldas, A., Kroonmaa, A., Laanoja, R., 2013. Keyless signatures' infrastructure: how to build global distributed hash-trees. In: Proc. of the Nordic Conference on Secure IT Systems, pp. 313–320.
Buldas, A., Laanoja, R., Truu, A., 2014. Efficient quantum-Immune keyless signatures with identity.. IACR Cryptol. 321. ePrint Archive
Casey, E., 2011. Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet. Academic Press, Waltham, MA.
Castro, M., Liskov, B., 2002. Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. 20 (4), 398–461. doi:10.1145/571637.571640.
Cucurull, J., Puiggalí, J., 2016. Distributed immutabilization of secure logs. Lect. Notes Comput. Sci. 9871 LNCS, 122–137. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)
Dennis, R., Owenson, G., Aziz, B., 2016. A temporal blockchain: a formal analysis. In: Proceedings - 2016 International Conference on Collaboration Technologies and Systems, CTS 2016, pp. 430–437. doi:10.1109/CTS.2016.80.
Guardtime, 2018. KSI Technology | Industrial Scale Blockchain | Guardtime.
Jämthagen, C., Hell, M., 2016. Blockchain-based publishing layer for the keyless signing infrastructure. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), pp. 374–381. doi:10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0072.
Ma, D., Tsudik, G., 2009. A new approach to secure logging. ACM Trans. Storage 5 (1), 1–21. doi:10.1145/1502777.1502779.
Menges, F., Böhm, F., Vielberth, M., Puchta, A., 2018. Introducing DINGfest: an architecture for next generation SIEM systems. In: SICHERHEIT 2018. Gesellschaft für Informatik e.V., pp. 257–260.
Mozilla Corporation, 2018. The Rust Programming Language.
New, D., Rose, M., 2001. Reliable Delivery for Syslog. Technical Report. IETF.
Onieva, J.A., Lopez, J., Zhou, J., 2009. Secure multi-party non-repudiation protocols and applications. Advances in information security. Springer, New York, NY.

Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S., 2007. A design science research methodology for information systems research. J. Manage. Inf. Syst. 24 (3), 45–77. doi:10.2753/MIS0742-1222240302.

Rasjid, Z.E., Soewito, B., Witjaksono, G., Abdurachman, E., 2017. A review of collisions in cryptographic hash function used in digital forensic tools. In: Procedia Computer Science, pp. 381–392. doi:10.1016/j.procs.2017.10.072.

Schneier, B., Kelsey, J., 1999. Secure audit logs to support computer forensics. ACM Trans. Inf. Syst. Security 2 (2), 159–176. doi:10.1145/317087.317089.

Shekhtman, L.M., Waisbard, E., 2018. Securing log files through blockchain technology. In: Proceedings of the 11th ACM International Systems and Storage Conference. ACM, New York, NY, USA, p. 131. doi:10.1145/3211890.3211921.

Sutton, A., Samavi, R., 2017. Blockchain enabled privacy audit logs. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 645–660. doi:10.1007/978-3-319-68288-4_38.

Tanenbaum, A.S., Van Steen, M., 2014. Distributed Systems: Principles and Paradigms. Prentice-Hall.

Venter, H.S., Eloff, J.H.P., 2003. A taxonomy for information security technologies. Comput. Security doi:10.1016/S0167-4048(03)00406-1.

Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J., Stein, J., 2015. Building a replicated logging system with Apache Kafka. Proc. VLDB Endowment 8 (12), 1654–1655. doi:10.14778/2824032.2824063.

Benedikt Putz studied at the University of Augsburg, University of Oulu and the University of Regensburg, where he received his Master of Science degree with Honors. Currently he is a research assistant at the Department of Information Systems at the University of Regensburg, Germany. His research concerns distributed ledger and blockchain systems, with a focus on applications of the technology in information systems security.

Florian Menges received both the Bachelor of Science and Master of Science degree from the University of Regensburg, Germany. Currently he is a research assistant at the Department of Information Systems at the University of Regensburg, Germany. His research interests include threat intelligence with a focus on sharing and reporting intelligence data, storage strategies for intelligence data as well as anonymization techniques and incentivizing the sharing and reporting of incident data.

Günther Pernul received both the diploma degree and the doctorate degree (with honors) from the University of Vienna, Austria. Currently he is full professor at the Department of Information Systems at the University of Regensburg, Germany. Prior he held positions with the University of Duisburg-Essen, Germany and with University of Vienna, Austria, and visiting positions the University of Florida and the College of Computing at the Georgia Institute of Technology, Atlanta. His research interests are manifold, covering data and information security aspects, data protection and privacy, data analytics, and advanced data centric applications.

2 RQ2: Protecting DLT applications from threats

2.1 Trust Factors and Insider Threats in Permissioned Distributed Ledgers - An analytical study and evaluation of popular DLT frameworks [P5]

Status:	published
Publication:	Transactions on Large-Scale Data- and Knowledge-Centered Systems
Submitted:	25 July 2019
Revised:	06 September 2019
Accepted:	29 October 2019
Citation:	Benedikt Putz and Günther Pernul. Trust Factors and Insider Threats in Permissioned Distributed Ledgers. Transactions on Large-Scale Data- and Knowledge-Centered Systems XLII, (2019), 25–50.

Journal Description: The LNCS Transactions on Large-Scale Data- and Knowledge-Centered Systems focuses on data management, knowledge discovery, and knowledge processing, which are core and hot topics in computer science.

Trust Factors and Insider Threats in Permissioned Distributed Ledgers
An Analytical Study and Evaluation of Popular DLT Frameworks

Benedikt Putz(✉) and Günther Pernul

Department of Information Systems, University of Regensburg, Regensburg, Germany
{benedikt.putz,guenther.pernul}@ur.de

Abstract. Permissioned distributed ledgers have recently captured the attention of organizations looking to improve efficiency, transparency and auditability in value network operations. Often the technology is regarded as trustless or trust-free, resulting in a false sense of security. In this work, we review the various trust factors present in distributed ledger systems. We analyze the different groups of trust actors and their trust relationships to the software layers and inherent components of distributed ledgers. Based on these analyses, we investigate how insiders may conduct attacks based on trust in distributed ledger components. To verify practical feasiblity of these attack vectors, we conduct a technical study with four popular permissioned distributed ledger frameworks: Hyperledger Fabric, Hyperledger Sawtooth, Ethereum and R3 Corda. Finally, we highlight options for mitigation of these threats.

Keywords: Trust frameworks · Distributed systems security · Distributed ledger technology · Insider threat

1 Introduction

Distributed ledger technology (DLT) offers great potential to decentralize operations in collaborative business networks and may even enable new business models [46]. Benefits include cost reduction and increased transparency in information sharing between organizations. However, great potential also entails great risks and potential security issues. Recent reviews regarding the future of blockchain technology have pointed out the need to study security and trust aspects of DLT [13,51].

Blockchain and DLT are often described as trustless or trust-free alternatives to currently established centralized systems (see [29,30,36]). In this work, we take a closer look at the usage of permissioned distributed ledgers and examine whether it can really be considered "trustless". The term "trustless" originates from the decentralization of control in distributed ledger networks [29], which aims to replace trusted third parties. The goal of this work is to establish a framework for reasoning about trust elements in permissioned distributed

A. Hameurlain and R. Wagner (Eds.): TLDKS XLII, LNCS 11860, pp. 25–50, 2019.
https://doi.org/10.1007/978-3-662-60531-8_2

ledgers. These trust elements can also be exploited by insiders, who are aware of them and in control of crucial components of the trust system.

Insider threats are a tough cybersecurity problem, which remains difficult to detect and prevent due to abuse of legitimate access permissions by the attacker. According to the roadmap of cybersecurity research by the US department of Homeland Security, insider threats are one of the "hard problems" of information security research [50]. Similarly, the European cybersecurity agency ENISA's threat landscape report lists insider threats among the top 10 information security threats, with 77% of companies' data breaches caused by insiders [23].

Insider threats are particularly relevant for distributed ledgers operated by a network of independent organizations. These networks are called *permissioned*, since they are operated by a restricted set of authenticated member nodes. In this scenario, intra-organizational insiders are supplemented with external insiders [25] from other organizations, who also have access to information shared on the network. According to a recent survey on enterprise adoption of DLT, there are at least 50 corporations with valuations larger than $1 billion looking to implement DLT to trade digital assets [11]. Many of these are financial institutions looking to trade high-value assets, leading to an attractive target for insider attacks.

To appropriately assess trust in distributed ledgers, our trust definition is based on software trust as defined by Amoroso and Watson [3]: *"Software trust is the degree of confidence that the software will be acceptable for 'one's needs'. It is established after one has become convinced, presumably based on a meaningful set of information, that the software does not include flaws that will prevent it from meeting its requirements."*

Besides trust in software components, the second form of trust is related to assessments of the human agents that collectively control the distributed system (hereafter referred to as trust actors). We follow Gambetta's definition of trust [28]: *"Trust (or, symmetrically, distrust) is (...) the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (...) and in a context in which it affects his own action."*

The remainder of this work is structured as follows: In Sect. 2 we give a short overview of related work concerning trust and insider threats with regard to distributed ledgers. Subsequently, we analyze trust actors, layers and components of permissioned distributed ledgers in Sect. 3. Based on our assessment of trust factors we identify relevant attack vectors for the different groups of insiders in Sect. 4. In Sect. 5 we perform a threat analysis for 5 popular distributed ledger frameworks, examining how insiders might exploit these vectors in practice. Finally, we wrap up by giving recommendations for future research in Sect. 6 and summarize our results in Sect. 7.

2 Related Work

While research on trust in blockchain systems is still scarce, the Global Blockchain Benchmarking Study by the University of Cambridge points out that blockchains always require some degree of trust [32]. Recent blog posts have highlighted trust factors in public permissionless blockchains [47]. Permissioned blockchains rely on similar trust primitives: trust in application code, network/cryptographic protocols and hardware. We aim to expand upon these notions by exploring the trust factors in more detail.

Overall, only partial aspects of trust in blockchain networks have been studied. Locher et al. [38] create a formal model to examine whether a distributed ledger may fully replace a trusted third party. In the process, they also evaluate previously proposed use cases of DLT that still require trust in other organizations and third parties. Hawlitschek et al. [30] review the conceptualization of trust in the blockchain environment. They argue that it is difficult to assess whether a system is actually trust-free or not. Correspondingly, another study claims that blockchain shifts trust from central authorities towards algorithms [39]. However, for this shift to be successful, the algorithms need to be trusted. Smart contracts represent the application-level algorithms, and their control flow immutability and independence of third parties have been shown to be lacking [26]. In addition to algorithmic properties, researchers have also studied user trust of different stakeholder groups from an HCI perspective [45]. In summary, research has established the existence of trust factors that contradict the claim of a trust-free system [30], but a comprehensive model of trust relationships is still missing.

Despite the severity of insider threats as pointed out by government agency assessments [23], research regarding insider threats in distributed ledger consortia is still scarce. Numerous surveys on the security of blockchain systems have been carried out [17,35], but none of them have focused on insiders in particular. We intend to fill this research gap and provide direction for future research.

In particular, we go beyond existing work by presenting a novel model of trust actors, their relationships and trusted DLT software components. We use this model to derive insider threats that organizations face when implementing permissioned DLT. By analyzing frameworks popular in both industry and research, we show that these attacks are applicable in practice. To protect against these threats, we outline technical and organizational options to mitigate the insider threats at hand.

3 Trust Factors in Permissioned Distributed Ledgers

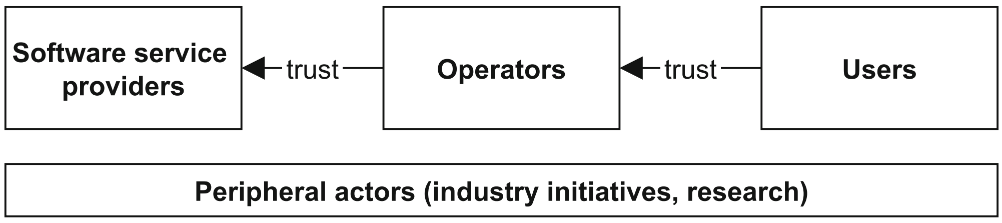

Software service providers
trust
Operators
trust
Users
Peripheral actors (industry initiatives, research)

Fig. 1. Trust actors and relationships.

As noted by other researchers, the requirement for trust does not disappear simply by employing a distributed ledger. Instead, trust shifts from trust in other organizations to trust in the technology and its operation [38]. In this section we focus on analyzing the different components of a distributed ledger system to establish trust actors, layers and components.

Before examining the trust factors in detail, trust actors need to be identified. There are four types of actors in the DLT ecosystem, three of which directly contribute to trust relationships in a consortium: software service providers, operators and users [32]. Peripheral actors (i.e. industry initiatives and researchers) do not directly interact with consortium networks, as they are not involved with building or operating DLT software. Nevertheless, they contribute by developing standards, methods and paradigms to solve current technical challenges [32].

An overview of the resulting trust hierarchy is shown in Fig. 1. **Software service providers** (SSPs) develop the software components of a distributed ledger consortium. They are trustees responsible for creating trust in the technology by developing secure and reliable applications. **Operators** represent distinct groups of actors responsible for running the distributed ledger overlay network and applications built on top of it. They act as both trustors and trustees: they trust the chosen DLT software to operate as expected and are also trusted by their users to provide reliable operations. Finally, **Users** place their trust in these applications and rely on them to work as advertised, without knowledge of the lower layers.

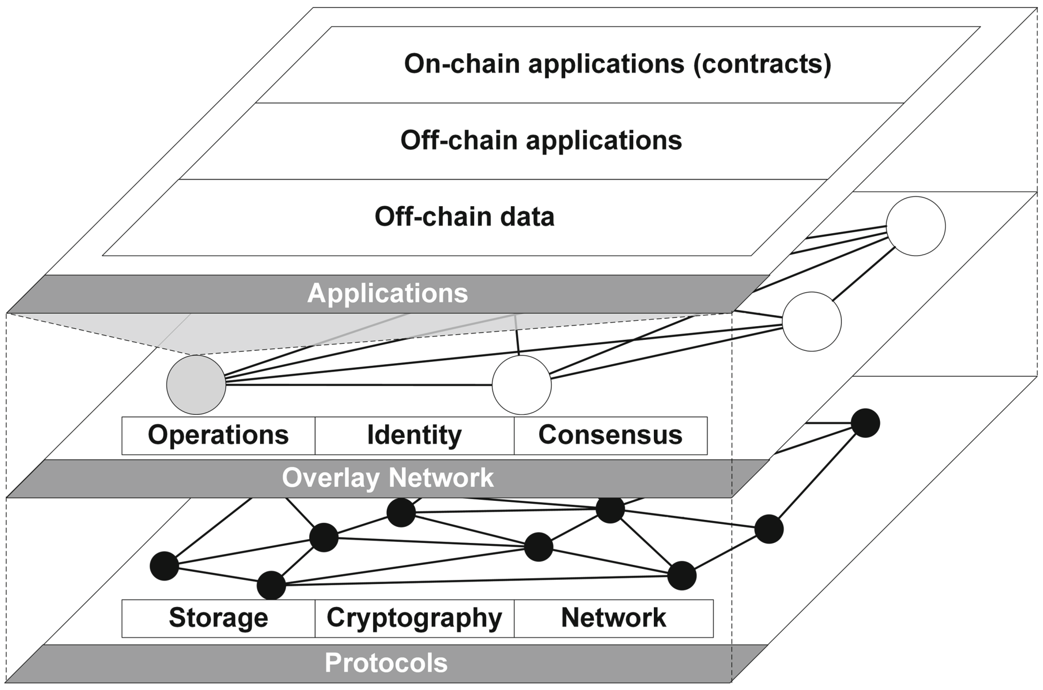


Fig. 2. Distributed ledger software layers and components.

The trust actors in Fig. 1 interact using a permissioned distributed ledger network, which consists of several software layers. Figure 2 shows the layers and the software components on each layer. They are derived from the three-layer view of Component Based Systems: platform, middleware and application [43]. The platform in this case consists of various underlying *protocols* responsible for storage, cryptography and network communication. Also part of the platform, but out of scope for this work, are the operating system and hardware layers. The middleware is represented by an *overlay network*, which provides configurable functionality for operations, identity management and distributed consensus. The *applications* layer provides replicated application logic (on-chain) and external logic and data (off-chain). These off-chain applications integrate with the framework by reading/writing data through its APIs. Each of the components within these layers requires software trust: it should be working correctly and not be maliciously exploitable. Since layers are built on top of each other, software bugs or vulnerabilities may propagate upwards to affect higher layers. In the following subsections we elaborate in detail on the layers' components and how they are involved in creating trust in the distributed ledger.

Figure 3 integrates trust actors from Fig. 1 with the layers from Fig. 2 by illustrating which trust actors govern each system layer. While software service providers are involved in the development of all three layers, operators do not interact with the underlying protocols. They merely configure the network framework and develop applications on top of it. Meanwhile, users only interact directly with the application layer.

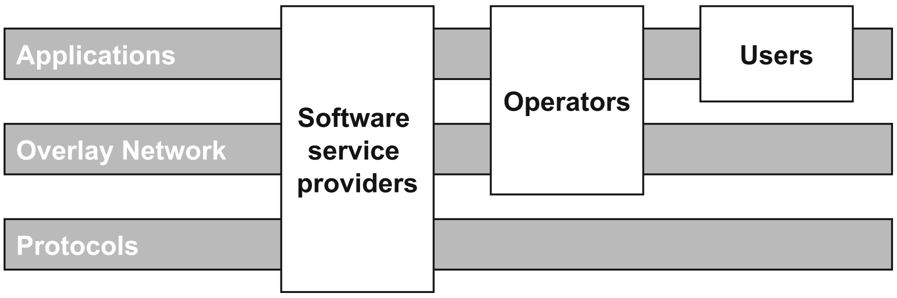


Fig. 3. Intersection of trust actors and layers.

3.1 Protocols

Storage, cryptographic and network protocols carry out the low-level tasks instructed by the distributed ledger framework and its applications. For this reason, they form the trust basis of the network.

Storage. A key property of blockchain-based systems is the goal of maintaining immutability of the underlying chain of blocks. Transaction and block metadata are stored in relational or key-value databases locally on each node. Replication integrity is assured through distributed consensus. The claimed immutability is a key factor in enabling trust in the technology, but it only holds if storage and network protocols can be trusted.

Cryptography. A manifold of cryptographic protocols are involved in distributed ledger operation. They include:

– hash functions for integrity assurance (hash chains, Merkle trees and proofs)
– public key cryptography (authentication of consensus protocol messages and user-submitted transactions)
– zero knowledge proofs (privacy-preserving transactions)
– symmetric encryption (on-chain confidentiality)

All of these protocols are trusted to not have design or implementation flaws. Many frameworks assemble their cryptography from a variety of sources, including standard libraries, external libraries and custom implementations (see Sect. 5). While some developers such as the Hyperledger open-source project perform third-party security audits [33], even the most diligent audits may miss vulnerabilities. Operators and users must trust protocol design and implementation, often without the ability to verify due to lack of cryptographic expertise.

Network Protocols. A fundamental trust factor for distributed ledger node communication is untampered operation of the underlying network. Distributed ledger networks are overlay networks, so they rely on P2P routing algorithms and message dissemination protocols for communication. Since all peers are equal, any single peer may cause disruption in the network by sending anomalous or malicious traffic. This may cause unexpected behavior and violate the aforementioned trust assumption.

3.2 Overlay Network

A distributed ledger network is a permissioned overlay network that consensually maintains a replicated ledger. In this overlay network, independent operators deploy a software framework previously agreed upon (i.e. Hyperledger Fabric). There are several tasks that each operator is trusted to fulfill by other participants: carry out operations tasks, maintain identity and access privileges and participate in consensus. The network layer also provides virtualization capabilities for replicated deterministic application execution in the application layer.

Operations. These independent organizations trust each other to perform node setup and operation without malicious manipulation. While there is some fault tolerance built into distributed ledgers (see Sect. 3.2), more than two-thirds of operators must behave honestly if byzantine-fault tolerant protocols are used.

Operation includes consensual admission/removal of members, on-chain application upgrades and framework upgrades. For the latter, all operators must agree on a coordinated time for system maintenance. Provided that all partners agree on the schedule, some partners might not be able to upgrade successfully [20]. Even if the failure to upgrade is not of malicious intent, it might pose considerable challenges to all involved parties, like setting up a new network and migrating data. Overall, the process requires significant trust in other organizations that cannot be mitigated by technology.

Identity and Access Control. Since the network is made up of independent organizations, each entity must be able to manage its users independently. This means that every organization must trust the others to properly manage identities and access rights. Since internal screening and job rotation processes are usually opaque to others in collaborative business networks [25], this can be considered blind trust.

Fundamental to permissioned distributed ledgers is an access control mechanism that ensures only authorized operators are part of the network. This is usually realized by assigning each node a public key, which is known to the other nodes and used to authenticate and secure communication. While admission/removal of node operators is based on majority consensus, other participants must be trusted to only accept legitimate new members. Additionally, compromise of a single node's credentials may undermine the trust assumption of a closed network.

Consensus. The consensus protocol is the distributed agreement protocol at the core of a permissioned distributed ledger, allowing all nodes to share a single replicated state. However, due to fundamental limitations underlying deterministic replicated state machines, less than one-third of participants may be malicious at the same time [12]. This limit means that operators must trust one another to act honestly and to not manipulate the consensus protocol.

3.3 Applications

Applications implement the business logic specific to each network. Next to on-chain smart contracts, off-chain applications and data are often required to implement all functionality and integrate with other enterprise systems.

On-Chain Applications. On-chain applications are generally referred to as smart contracts, although this depends on their implementation (see Sect. 5.7). They promise to replace trust through replicated and verifiable deterministic execution. Since both code and state can be inspected by anyone with access to the ledger, their execution is predictable to these parties. However, a number of smart contract vulnerability studies have shown that code is not always law and may be exploited to an attacker's advantage. For example, in 2016 a symbolic execution tool found almost half of all Ethereum contracts at the time to be vulnerable [40]. In another study, 2 out 5 deployed Ethereum contracts were shown to require trust in at least one third party, since parts of their control flow may be changed after deployment [26].

Off-Chain Applications. One example for off-chain applications are web applications for user interaction with the distributed ledger. Without a way to verify what is going on behind the scenes, users must blindly trust that the application does not manipulate any data sent through it. While this is also the case for

Table 1. Summary of distributed ledger trust components by layer.

Protocols	Overlay network	Applications
Storage	Operations	On-chain applications
Cryptography	Identity	Off-chain applications
Network	Consensus	Off-chain data

traditional web applications, a distributed system aiming to create trust should provide stronger guarantees.

Off-Chain Data. Full replication of the blockchain data structure mandates parsimony w.r.t. transaction sizes. Distributed ledger applications rely on off-chain storage solutions to manage larger data volumes. In fact, a recent study found that a majority of DLT operators only include hashes in on-chain transactions [32], which point to off-chain data and serve as integrity timestamps. Operators must trust their peers to maintain sustained availability, since off-chain data is not fully replicated. If off-chain data is access protected, the storage operator must also be trusted to maintain correct access privileges.

Besides referenced data, external data may also be needed as input for computation (i.e. currency exchange rates). Since external data sources must return deterministic results, distributed ledgers rely on trusted external content providers (oracles) - hereby reintroducing trust elements.

A summary of all identified trust components is shown in Table 1. Overall, the complexity of the DLT software layers results in a high degree of obscurity. It becomes increasingly difficult to verify correctness and security of the software stack. The trust actors (operators, users and software service providers) must trust both software components and each other to act as expected. In the next section, we focus on how insiders can exploit these trust assumptions.

4 Insider Threats

Given the aforementioned trust elements required for operating a permissioned distributed ledger network, insider attacks may pose a significant threat. With the emergence of business networks and blockchain consortia for data sharing, the partners' information systems infrastructures are no longer isolated environments with a protectable logical perimeter. Access to an organization's resources is implicitly simplified for outsiders that are part of the consortium. This is a direct consequence of sharing information with other organizations.

As a result, a holistic security model must also include actors from network partners. This group of threat actors is also known as **external insiders**. External insiders are characterized by having limited access to an organization's network as a result of some business relationship [25]. In a distributed ledger context, the relationship may be the result of a collaborative network with multiple organizations.

Table 2. Overview of insider threats and consequences.

Insider type	Threat	MO	DE	DI	DU
Software service provider	Vulnerability injection	x	x	x	x
Operator	Denial of service				x
	Data manipulation	x	x		x
	Credential compromise	x		x	x
	Malicious misconfiguration			x	x
User	Unauthorized operations	x			x
All	Vulnerability abuse	x	x	x	x
	Information leakage			x	

Subsequently, we describe the various insider threats to distributed ledgers by analyzing each group of trust actors. Irrespective of an insider's group, there are generally four types of consequences an insider might achieve in an attack [37]:

– Modification (MO)
– Destruction (DE)
– Disclosure (DI)
– Denial of use (DU)

Insiders may exist in any of the three groups of trust actors stated in Fig. 1. Depending on the group, there are different ways to exploit their privileges. Table 2 lists the major categories of insider threats and their consequences in distributed ledger consortia. While many of these threats are also applicable to existing information systems, distributed ledgers are particularly vulnerable due to the large number of software components and cross-organizational users.

Permanent modification or destruction of data are generally difficult to achieve with DLT due to built-in fault tolerance and replication. Nevertheless, collusion-based data manipulation or software vulnerabilities may cause data manipulation on all nodes. Disclosure of information and denial of use are the more likely consequences of an insider attack on a distributed ledger. They are significantly easier to accomplish and may be achieved with user or operator level permissions. We elaborate on these threats in detail hereafter.

4.1 Software Service Providers

If an insider is in the role of an internal software developer with full code access, there is significant threat potential for any of the four consequences to happen. Collins et al. [15] have surveyed a variety of methods that programmers acting as malicious insiders have used in the past. Common methods are code modification or injection of malicious code, causing vulnerabilities. Characteristic for this type of manipulation is a time delay between injection and impact of the attack, since software builds go through testing and deployment phases. In large software projects without strict code review procedures, these types of manipulations may easily go under the radar of other developers. Vulnerabilities can then be abused by the programmer or colluding operators/users. They may corrupt the integrity and availability guarantees that a distributed ledger provides, leading to manipulation or loss of information. Intentionally timed bugs may cause network unavailability. Backdoors (either for outsiders or insiders) could be inserted that lead to disclosure of confidential information.

The potential attack vectors depend on the developer's area of responsibility (protocol/framework/application level). Currently, distributed ledger protocols and overlay-level frameworks are often provided by open-source initiatives. If an open-source project is subject to peer-review and security audits, these parts of the software are unlikely to be affected by insider attacks threatening a single organization. However, applications built on these frameworks must be customized to specific business requirements—either by employees or third party developers. For this reason, the application level is more prone to software development insider threats. Business networks may collaboratively develop applications, which extends this attack vector to external insiders.

4.2 Operators

Overall operators have the largest number of insider attack options at their disposal, since they directly control a network node. Even though a single operator controls only one node, malicious behavior may have powerful denial-of-service effects on networks with few participants, as mentioned in Sect. 3.2.

In larger networks, operators could take advantage of their knowledge about the current consensus leader. A malicious operator may launch a targeted denial of service attack to cause network interruptions (see Sect. 5.6). Additionally, collusion of operators from other organizations (i.e. by external insiders) might result in denial of service, if it leaves the network unable to reach consensus. Generally, network operators have a common goal and should not be inclined to collude against others. This might change if goals shift, or partners feel that they contribute more to the network than they gain in return.

An insider might attempt to accomplish modification of stored data by coordinating an attempt to replace ledger data in collusion with other nodes. This type of attack is known as a 51%-attack for permissionless blockchains [35]. Generally this is only feasible if consensus is stochastic, while permissioned networks usually rely on deterministic consensus. Nevertheless, availability of off-chain data with low replication factors is still at risk.

System administrator insiders have access to all relevant credentials for node operation. These credentials can be leaked, or misused by adding/removing users or manipulating access control privileges at will.

Another way to subvert data integrity is configuration manipulation. Successful configuration manipulation requires collusion, since such changes need to be approved by a majority of operators in properly configured networks. Without automated punishment mechanisms, a single misbehaving node may however still cause temporary service disruption.

4.3 Users

Users have only a limited number of options for exploiting the distributed ledger network. Nevertheless, due to external insiders the number of potential attackers is higher than in intra-organizational applications. Improperly managed access rights for these users may enable leakage of confidential information.

Another attack vector are vulnerabilities in custom-developed contracts. They could enable insiders to carry out unauthorized asset transfers or even shut down an application. Insiders might have increased knowledge about application internals such as access to source code and technical documentation, enhancing their ability to discover programming flaws and carry out unauthorized operations.

It is important to note that threats are not strictly restricted to a specific group of trust actors. For example, operators may also impersonate users if they control the identity component. Conversely, users may gain operator-level privileges through improper access right management.

In fact, some threats are exploitable by any actor with access to the distributed ledger network. Whether intentional or inadvertent, vulnerabilities can be abused by any insider with the required knowledge and skills. Since all nodes of the network likely run the same software, remote code execution vulnerabilities may lead to irreversible manipulation or loss of data. Similarly, any participant with access to distributed ledger data may leak data to parties outside the network. The extent of information leaked depends on the insider's access privileges.

5 Insider Threat Analysis of Popular Frameworks

To demonstrate applicability of the identified insider threats to permissioned blockchain frameworks, we conduct a technical threat analysis of several popular blockchain frameworks. Frameworks were selected based on a survey of industry and research support conducted in July 2019. Regarding industry support, the Ethereum Enterprise Alliance (240 members), the Hyperledger project (249 members) and R3 (92 members and 294 partners) were the largest enterprise consortia working on open-source permissioned DLT frameworks. To gauge research interest, we searched several literature databases for mentions of permissioned distributed ledger frameworks. We used fulltext search since some framework names are ambiguous and might be used with a different meaning in a distributed systems context (i.e. Quorum). The search result counts totaling close to 1000 academic publications are shown in Fig. 4. As a result of this survey we decided upon the four frameworks detailed below.

We briefly describe each framework below and summarize the technological components in Table 4. To future-proof our analysis, we mainly analyze threats resulting from architectural design choices, which are unlikely to change in the future. We assume that operators strive for a secure configuration, which includes a byzantine-fault tolerant (BFT) consensus algorithm to prevent byzantine manipulations.

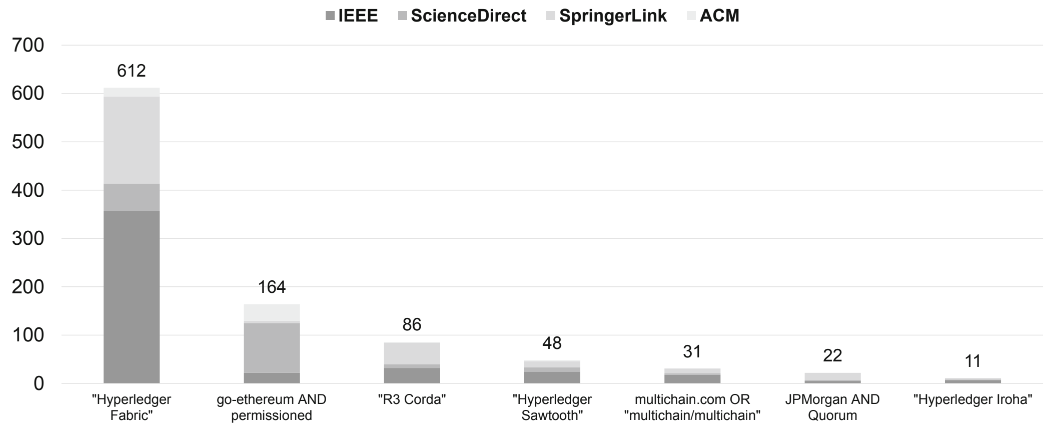


Fig. 4. Research popularity of distributed ledger frameworks (search result count by search term and academic database).

Hyperledger Fabric is a blockchain framework relying on a novel execute-order-validate architecture. This architecture was created to rule out source of non-determinism during consensus and improve performance [4]. We assume that the BFT-SMaRt consensus algorithm[1] [9] is used for ordering service consensus, since it is to our knowledge the only currently available BFT consensus module for Fabric.

Hyperledger Sawtooth [49] is a blockchain framework, which modularizes transaction processing with so-called transaction families. They include predefined families for permissioning and on-chain settings management. Consensus is also modular, but we assume that the Practical Byzantine Fault Tolerance (PBFT)[2] [12] module is used.

Ethereum [18,24] is a popular permissionless blockchain framework, which runs smart contracts written in the Solidity language in an isolated environment called the Ethereum Virtual Machine. The go-ethereum client can also be set up as a permissioned network with Clique Proof-of-Authority (PoA) consensus. PoA is a leader-based consensus protocol with stochastic consensus, which only provides eventual consistency as opposed to strong consistency provided by BFT algorithms. It only requires 50% for a consensus majority, trading consistency for availability.

R3 Corda [31,44] is based on a DAG data structure and only shares data with other nodes when needed. To prevent double spending, mutually agreed upon notary service clusters are used for consensus. We assume that BFT-SMaRt consensus is used among notaries, since it is the only built-in consensus algorithm which tolerates byzantine faults.

All of these frameworks have unique differences in their architecture and the way applications are built on them. While some threats are applicable to all frameworks, others apply only to specific frameworks due to architectural

[1] github.com/bft-smart/fabric-orderingservice.
[2] github.com/hyperledger/sawtooth-pbft.

Table 3. Mapping of insider threats to abused distributed ledger trust components.

Threat	**5.1**	**5.2**	**5.3**	**5.4**	**5.5**	**5.6**	**5.7**	**5.8**	**5.9**
Vulnerability injection		x		x		x	x	x	
Denial of service	▨	▨	▨	x		x			
Data manipulation	▨	▨	▨		x	x			x
Credential compromise	▨	▨	▨		x				
Malicious misconfiguration	▨	▨	▨	x	x	x			
Unauthorized operations	▨	▨	▨	▨	▨	▨	x	x	
Vulnerability abuse	x		x				x	x	
Information leakage	x				x				x

choices. Subsequently, we survey each framework's trust components and analyze where insiders may abuse architectural flaws.

Table 3 highlights which framework components that each type of threat exploits. The columns represent the subsections corresponding to the affected trust components. Corresponding to the trust actors that each threat applies to (Table 2) and their ability to access various trust layers (Fig. 3) some cells are marked gray (not applicable). Vulnerability injection does not result in a vulnerability abuse threat if the vulnerability can only be effectively abused by the software service provider. Hereafter, we explain per component how each threat may occur. Threats are italicized when they refer to a table entry.

5.1 Storage

The surveyed distributed ledger frameworks mostly rely on existing key-value databases for data storage (i.e. LevelDB and CouchDB). None of these databases offer encryption-at-rest, which means anyone with access to the database can read all historical data contained in the ledger. Corda marks the exception: it relies on relational databases, some of which offer encryption. Nevertheless, the database itself is an attractive attack vector for operator insiders, who may circumvent framework-level access control by directly accessing the underlying database *(information leakage).*

5.2 Cryptography

Currently, distributed ledgers almost exclusively use public key cryptography for authentication. The reviewed frameworks use NIST-recommended ECDSA curves in combination with SHA2 for digital signatures, with some also offering EdDSA and RSA. These algorithms are vulnerable to quantum attacks based on Shor's algorithm [6] *(vulnerability abuse).* Such attacks threaten the authenticity of transactions and network messages (see Sect. 5.5). If symmetric keys were encrypted using public key cryptography, this may also result in *information leakage.* Once quantum computers reach sufficient computational power, current ledgers will have to be rebuilt from scratch with new identity schemes. Instead of relying on a single cryptographic primitive, developers should instead adopt a more future-proof approach. For example, quantum-proof hash-based signature schemes use hash combiners, which remain secure if at least one of the input hash functions is secure [6]. In our review, only Corda offers such a scheme with SPHINCS256[3]. Still, no framework provides guidance for migration between signature schemes.

Due to the unique challenges of trust in distributed environments, some distributed ledger frameworks rely on new variants of cryptographic protocols with unproven implementations (especially novel non-interactive zero-knowledge proofs such as zk-STARKs [8]). While we are not aware of any cryptographic flaws in the reviewed permissioned frameworks, there are examples among permissionless blockchains. Recently, a zero-knowledge proof vulnerability in the permissionless blockchain Zcash was publicized, which had been kept secret by the development team for more than 11 months [48]. In this case the developer that discovered the bug did not have malicious intentions and worked on fixing the bug instead of exploiting it. But the incident shows that open-source code is not immune to longstanding hidden vulnerabilities. These may even be inserted into the code intentionally by members of the development team *(vulnerability injection)*. If discovered by malicious actors, they could be kept secret for continued exploitation. Overall, trust in the security of cryptographic protocols is not guaranteed and may be undermined at any time.

5.3 Network Protocols

The reviewed frameworks rely on different network protocols for node-to-node communication. The ZeroMQ protocol[4] used in Sawtooth and the AMQP protocol[5] used in Corda have experienced denial of service and remote code execution vulnerabilities in the past[6]. External insiders, who know about the underlying protocols, may abuse these vulnerabilities to cause damage to specific competitors in the network *(vulnerability abuse)*.

Consensus protocols require constant network communication between all involved nodes. For this reason, they are vulnerable to network-partitioning attacks such as Border Gateway Protocol (BGP) hijacking and Eclipse attacks [35]. These attacks have so far mainly been observed and studied on permissionless blockchains. In permissioned blockchains, manipulations of the routing protocol or network traffic interception can also lead to network partitions [22]. If none of the partitions are large enough to reach consensus, the network will stop processing incoming transactions (deterministic algorithms) or create competing forks (stochastic algorithms) [21]. For some consensus algorithms, these network partitions can even allow malicious double-spending transactions (see Sect. 5.6).

[3] sphincs.cr.yp.to.
[4] zeromq.org.
[5] www.amqp.org.
[6] cve.mitre.org.

Table 4. Overview of software components used in popular distributed ledger frameworks

	Hyperledger Fabric v1.4	Hyperledger Sawtooth v1.1	Go-Ethereum v1.9	Corda v4.1
On-chain contracts	Chaincode (Go, nodeJS, Java)	Transaction Processor (see below)	Smart Contract (Solidity)	CorDapps (Java)
Off-chain applications	Go, Java, nodeJS, Python	Go, Java, nodeJS, Python, C++, C#, Swift	Web3 (nodeJS)	Java
Off-chain data	–	–	Swarm	Oracles
Operations	ReST Operations Service, CLI	Settings TP, CLI	RPC CLI	RPC CLI
Identity	Membership Service Provider	Identity Transaction Processor	Accounts	Hierarchical PKI, Doorman Service
Consensus	Endorsements (custom), Ordering (Kafka, BFT-SMaRt)	Journal (PoET, Raft, PBFT)	PoA, IBFT	Notary (Raft, BFT-SMaRt)
Storage	LevelDB, CouchDB	LMDB	LevelDB, RocksDB	H2, Postgres, SQLServer
Cryptography	ZKP: idemix Signature: ECDSA P256/384, Hash: SHA256, SHA3 Encryption: AES	Hash: SHA256/512 Signature: libsecp256k1	Hash: Keccak Signature (using SHA256/384/512, AES): ECDSA P256, P384, P521, S256, bn256	Hash: SHA256 Signature (using SHA256/512, AES): RSA; ECDSA secp256r1, secp256k1; EdDSA-ed25519; SPHINCS256
Network	GRPC, Gossip[a]	ZeroMQ	devP2P[a]	AMQP

[a]custom protocol

5.4 Operations

Regarding operational tools, the frameworks offer little in terms of monitoring capabilities. Only command-line interfaces (CLI) and transaction types for on-chain settings (Hyperledger Fabric and Sawtooth) are offered to retrieve metrics and update settings. Without significant effort by the operators, this may lead to manipulations of configuration settings going undetected *(malicious misconfiguration)*. Additionally, intentional manipulations of consensus network traffic are nearly impossible to detect without proper monitoring. For example, TCP or UDP flooding attacks reduce transaction throughput to a small fraction of peak throughput [14]. Lack of monitoring facilities effectively allows operator (external) insiders to control network throughput by launching attacks when desired *(denial of service)*.

Despite their initial immutability, smart contracts can be upgraded in all surveyed frameworks[7]. If only bytecode is available for inspection, there is no

[7] Ethereum only allows upgrades if the contract has been set up in a modular way.

easy way to tell what part of the contract was changed. Since smart contracts may contain vulnerabilities or require feature extension, upgrades cannot be regarded as unusual per default. Individual operator insiders may abuse this fact by upgrading a contract with malicious functionality *(vulnerability injection)*. Requiring signatures of multiple operators for a successful upgrade is a potential mitigation, but the contract needs to be set up with multiple owners for this to apply (Ethereum, Fabric). Corda requires all participants that share a state to sign contract upgrades, but contract signature constraints also permit custom rules that require less signatures for an upgrade [44]. Sawtooth offers no mechanism for coordinated upgrades. Transaction processors run as independent processes next to the validator network and are upgraded by each operator individually.

5.5 Identity and Access Control

All permissioned networks must admit identities through some form of a gatekeeper. Distributed ledger frameworks attempt to decentralize admission of validating nodes by voting on new members. This does not apply to users, who must receive a certificate through a validating node or its certificate authority. In Ethereum and Sawtooth, there is no certificate infrastructure integration: users either create accounts themselves or request them from a node administrator. However, account pseudonyms need to be mapped to real-world identities for many applications. A lack of certificates complicates permission revocations, which then need to be performed on the application level. Lack of a single source of truth for identity also leads to excessive access rights over time, which enable insider abuse [27]. A potential consequence of unchecked access rights is *information leakage*.

Conversely, Hyperledger Fabric and Corda rely on traditional hierarchical PKIs with a root authority. In these systems, each node operates its own certificate authority. Certificates are then shared across the network through a federation service. As a result, each validator node recognizes the identities issued by its peers. From an insider perspective, this means that any employee with access to the gatekeeper of a participating organization can create valid identities for the entire network. By subverting the local certificate authority, operator insiders may replace associated node's certificate and impersonate it *(credential compromise)*. Accordingly, a collusion between CA operators can subvert multiple node identities and overtake the network, thus enabling *data manipulation*.

Fundamentally, the identity component relies on the underlying cryptographic signature protocols. If a cryptographic primitive is broken, credentials can be forged by issuing fake signatures. Fake signatures in turn enable *data manipulation* and *malicious misconfiguration* through malicious transactions.

5.6 Consensus

The surveyed permissioned distributed ledger frameworks rely on crash fault-tolerant (CFT) or byzantine fault-tolerant (BFT) algorithms. CFT algorithms do not tolerate any malicious activity and are built only to tolerate crashes [10]. As a result they are prone to manipulation by any one operator and not well suited for usage in semi-trusted environments like business networks. Despite this, most frameworks in our survey recommend CFT protocols and mark BFT consensus implementations as experimental.

If operators use BFT algorithms, up to f malicious nodes among $n = 3f + 1$ total nodes are tolerated without ceasing operation. Byzantine failures encompass all possible failure modes of a system. The performance of most BFT algorithms is however heavily impacted by the presence of failures and no consensus is reached with more than f failures. As a result, a single operator can significantly decrease throughput in PBFT-based networks with $n < 7$ independent nodes $(3f + 1 = 7 \mid f = 2)$ by flooding the network with messages [14]. Collusion between two operators can even shut down network consensus and prevent new transactions *(malicious misconfiguration)*. Therefore, smaller permissioned networks are especially at risk of *denial of service* by a minority of participants. In addition to flooding-based denial of service, malicious consensus leaders can also degrade performance in most BFT consensus protocols [5].

The PoA algorithm used by permissioned Ethereum networks is vulnerable to the Attack of the Clones [22]. In the attack, a single malicious node can double spend with high probability. By cloning itself and intercepting messages, a network partition is created. The victim partition is deceived by submitting a conflicting transaction to the other partition, which is later accepted as the canonical transaction *(data manipulation)*. Based on the authors' assessment, the only viable countermeasures are switching to a BFT algorithm or requiring a two-thirds majority instead of the current 50%.

In addition to protocol flaws, *vulnerability injection* in the consensus protocol implementation may lead to nodes accepting invalid or malicious transactions. This could be abused by SSP insiders to circumvent on-chain access permissions and transfer assets or tokens.

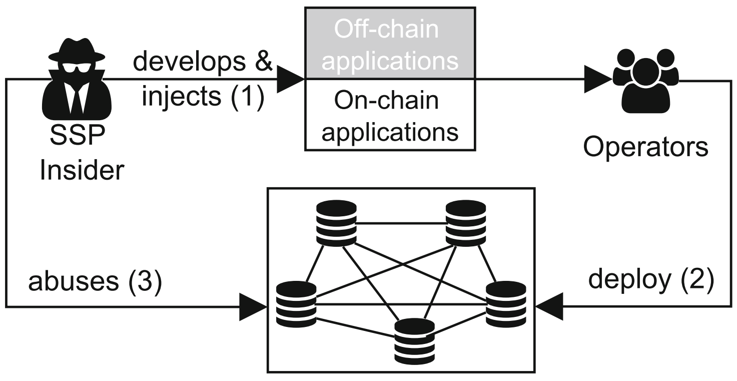


Fig. 5. Illustration of injection and delayed abuse of a vulnerability by a SSP insider.

5.7 On-Chain Applications

Transparency is an often-cited advantage of smart contracts. In the surveyed frameworks, contract code is rarely transparent to all operators, and never to users. In Hyperledger Fabric, chaincode source code is only known to the peers specified in its endorsement policy. For Sawtooth, bytecode is deployed when using the Seth (Solidity), Sabre and Java SDKs. For transaction processors based on the Python and nodeJS SDKs the source code is transparent, since they are interpreted languages. In Ethereum, only compiled Solidity bytecode is visible to blockchain node operators. Corda's CorDapps are only shared by peers concerned with the application, who need to compute the state changes for notary consensus. A lack of transparency can lead to undetected manipulations, for example through covert contract upgrades.

To demonstrate how this might occur in practice, Fig. 5 illustrates the three steps of *vulnerability injection* and subsequent *vulnerability abuse*. First, the SSP insider injects a vulnerability into an on-chain or off-chain application (1). With the next scheduled operational software upgrade, operators deploy the vulnerable version of the software to the production network (2), permitting the insider to abuse the vulnerability (3).

For Hyperledger Fabric, the two most popular SDKs on GitHub are based on nodeJS and Go. Both languages allow package imports from public version control sites such as GitHub. This method could be abused by a software service provider to conceal malicious functionality in the chaincode or insert a backdoor. By changing the code of a self-controlled dependency, the developer gains the ability to manipulate dependent code sections, while obscuring the changes from the client. Additionally, existing vulnerabilities in packages may be knowingly included by a SSP insider to be exploited later on. Due to the large number of transitive dependencies, the nodeJS dependency management system npm is especially prone to attacks based on existing vulnerabilities [19]. For Ethereum smart contracts, many vulnerability classes are known [35] due to public scrutiny and open source application bytecode. For Sawtooth and Corda there are not many production deployments yet, so to the best of our knowledge we are not aware of any vulnerabilities.

To summarize, chaincode and smart contract vulnerabilities may manipulate the output state of a contract *(data manipulation)*, prevent consensus by introducing non-determinism *(denial of service)*, and leak secrets by sending confidential data to parties outside the network *(information leakage)*. Additionally, users may be able to conduct *unauthorized operations* due to a smart contract permission management vulnerability.

5.8 Off-Chain Applications

Production deployments of DLT must include a client software, since direct interaction with a blockchain node is not user-friendly and requires command line skills. This client software relies on Software Development Kits (SDKs) published by software service providers. Table 4 shows the diverse programming languages that these SDKs use. For SDKs using package managers, the same attack vector from Sect. 5.7 applies *(vulnerability injection)*. SSP insiders may abuse SDKs or client software to hijack user identities and thus gain access to the distributed ledger network *(credential compromise)*.

5.9 Off-Chain Data

For referenced off-chain data, the reduced replication factor exposes data availability to collusion attacks. Depending on the replication factor r of items stored off-chain, r operators may collude to irreversibly delete a key-value pair stored off-chain *(data manipulation)*.

Off-chain events providing external data are not natively supported by the surveyed frameworks. The requirement for code determinism runs counter to uncertain responses from external sources. Consequently, external data is often integrated via trusted applications that attempt to guarantee response integrity. These applications are referred to as validation oracles [52]. They act as automated arbitrators that sign transactions referencing external data on demand. Corda is the only framework that provides built-in integration with centralized oracle services for this purpose. They are accepted as an authoritative source of data by a set of peers. An insider may compromise that service and manipulate transactions at will, without needing access to the distributed ledger *(data manipulation)*. Depending on the application, such attacks can be hard to discover, since the oracle service is not transparent to all operators. Alternative proposals that avoid relying on a centralized provider include secure hardware architectures such as Town Crier [53], and cryptocurrency-based decentralized blockchain oracles such as Astraea [1].

6 Mitigations and Future Research

The previous sections have shown how various types of trust actors in a distributed ledger system may abuse trust components and carry out insider attacks. Based on these insights, we now elaborate how DLT adopters can better assess which components they trust, and how they can mitigate resulting insider threats.

6.1 A Realistic View of Trust in Distributed Ledgers

Instead of regarding blockchain as "trustless", DLT adopters should be aware of the technological components that their trust relies on. First and foremost, software trust management processes should be established to ensure that trust is warranted. The inherent failure modes and consequences should be integrated into organizational risk management.

Regarding trust in the various software components of distributed ledgers, software trust research has established trust principles and an ordered set of classes for software trust measurement [3]. The classes range from Untrusted

(T0) to Trusted (T5) and require a progressively larger set of trust principles to be fulfilled. Classes T4 and T5 aim to prevent malicious activity and could be used to certify components for inclusion in trusted distributed ledgers.

To reduce the implicit trust resulting from allowing others to manage identities and access rights, trust-based distributed access control models could be used. Such frameworks include risk assessment processes that dynamically adapt to users' behavior [7]. Additionally, next generation decentralized blockchain-based identity management could enable consensus-based trust in external users, instead of relying on federated membership schemes.

To increase trust in smart contracts, a number of approaches have been proposed. Business and legal smart contract specification languages based on formal reasoning can help reduce ambiguity for programmers and lock down edge cases [2].

To make user interaction with distributed ledgers more transparent, "decentralized applications" (DApps) may be used. DApps are web applications relying solely on an on-chain application backend. DApp frontends should be distributed as client-only applications, with code only served from, but not executed on a centralized web server. This ensures that code execution is fully transparent to end-users. Transaction submission to the blockchain can be explicitly authorized using open source browser extensions (i.e. MetaMask [42]).

6.2 Insider Threat Mitigation

Techniques for insider threat mitigation have been studied extensively in the past [16]. They require an interdisciplinary mitigation approach, combining insights from computer science, psychology and other fields. Correspondingly, mitigation techniques can be classified as technical or organizational. We have analyzed mitigation techniques in the literature and applied them to the threats mentioned in Table 2. The result of this analysis is shown in Table 5. We emphasize technical measures, but organizational mechanisms are sometimes required and often beneficial.

To counter injection of vulnerabilities or flaws by **software service providers**, *code review* processes help assure sufficient oversight and reduce the probability for malicious activity to go unnoticed. From a framework perspective, *transparency* should be assured by embedding on-chain application code into the distributed ledger framework. Participants should be able to retrieve source code for a contract at any given time. Re-compiling source-code from a repository is too time-consuming and error-prone to serve as a verification method for smart contract bytecode.

Additionally, dedicated *vulnerability scanners* exist specifically for distributed ledger on-chain applications [40]. Whether these scanners are suitable for detecting insider-induced intentional manipulations of program flow has yet to be shown. Future empirical research may also analyze specific programming techniques that insiders use to attack distributed ledgers, similar to the study conducted by Collins et al. [15].

Table 5. Overview of potential mitigations for insider threats. O: organizational, T: technical.

	Threat	Mitigation	O/T
Software service provider	Vulnerability injection	Software trust management	O
		Code review and transparency	O, T
		Vulnerability scanners	T
Operator	Denial of service	Legal agreements	O
		Robust consensus algorithms	T
	Data manipulation	Automated detection and punishment	T
	Credential compromise	Credential revocation mechanisms	T
	Malicious misconfiguration	Configuration integrity checks	T
User	Unauthorized operations	Activity monitoring	O, T
		Anomaly detection	T
All	Information leakage	Granular Identity and Access Management	T
		Granular encryption	T
	Vulnerability abuse	Software update management	T

Intentional *denial of service* caused by a collusion of **operators** may be mitigated through legal agreements and incentives. While research on collusion in distributed ledger networks is still scarce, trust-based reputation algorithms could help. They may prevent collusion attempts or at least minimize their impact on network availability by punishing colluding peers. In the past, reputation systems based on peer-to-peer networks have also faced the issue of collusion [41]. Future research could thus transfer insights on collusion prevention from peer-to-peer reputation systems research and related areas to distributed ledgers.

Regarding consensus attacks targeting network availability and throughput, *robust consensus algorithms* such as RBFT [5] can help. Robust protocols sacrifice some throughput compared to traditional algorithms [14], but maintain high availability and throughput regardless of attacks.

Activity monitoring tools and corresponding organizational processes assist with timely detection of data manipulation. The threats listed in this work can serve as guidance for activities to monitor. To prevent insider abuse by unchecked node administrators, monitoring should be part of IS security management and organizationally separated from operational IS administration. Similarly, certificate authority and DLT node should be managed by separate entities. If any entity attempts to manipulate node settings, automated *configuration integrity* *checks* should raise alerts in security monitoring units across the network. In case of external insiders, attack attempts should be punished either through legal agreements or a built-in incentive system.

Transaction *anomaly detection* could help spot unauthorized operations. If anomalies are detected in time, further abuse of permissions can be prevented. For traditional database systems, tools have been developed that automatically determine profiles of normal activity based on application profiles [34]. Similar security analytics tools can be developed to detect anomalous smart contract transactions.

One of the main countermeasures for information leakage by **users** is *granular identity and access management* (IAM) [27]. If users cannot send transactions or access ledger data without first being authorized by an application owner, the attack surface becomes significantly smaller. For authorized users, automated detection and timely removal of access privileges helps limit permission buildup and impact of insider attacks. Still, it remains challenging to ensure that each operator of the network keeps its individual permissions and access rights updated. Future work could examine how to enforce granular IAM network-wide, for example through automated checks or incentives.

Another tool to limit malevolent information disclosure is *granular encryption* of data, ensuring that users are only able to view data they need to access. Organizations must correctly decide where to encrypt data on the application level. Additionally, they should be parsimonious regarding confidential data stored on the ledger. Symmetric encryption keys may eventually be leaked, but on-ledger data cannot be deleted.

Overall, we observe that a holistic security monitoring concept is necessary for each organization participating in a distributed ledger network. A set of standardized monitoring metrics is a necessary prerequisite to detect manipulations of the various trust components. Due to unique differences in distributed ledger architectures, the metrics need to be customizable to the specific application context. One metric should be vulnerabilities in framework components, with automated *software update processes* attempting to minimize the number of vulnerabilities. Future work should focus on creating and evaluating such a security management framework for permissioned distributed ledger networks.

7 Conclusion

While distributed technology offers great benefits, organizations planning to take advantage of it should be aware of the trust relationships they enter. In this work, we established key trust actors and components for distributed ledgers to provide a better understanding of hidden trust factors and security risks. On the one hand, software trust in the components of a distributed ledger system is a key factor. If there are vulnerabilities or bugs, trust assumptions may be violated with grave consequences. Additionally, operators must still trust one another to some degree to cooperatively run a network. If this trust turns sour during operation, the distributed ledger network may become subject to denial of service or collusion attacks.

These attacks may be especially severe if carried out by insiders. They possess unique access to organizational resources that may facilitate subversion of distributed ledger trust assumptions. Insiders involved in application development may willingly inject vulnerabilities or malicious code. Node administrator insiders have elevated access rights to credentials and configuration. Malicious manipulation of these components may result in denial of service for the entire network. Modification or destruction of data are also possible in some cases (see Sect. 4), despite the integrity guarantees that distributed ledgers normally provide. Both internal and external insiders may leak data or abuse vulnerabilities in the distributed ledger software stack.

Due to the current lack of productive deployments of distributed ledger networks, this work focused on analyzing the potential impact of insider attacks from a theoretical perspective. To reinforce these claims, we elaborated on how insiders may exploit the architecture of popular distributed ledger frameworks. Future research may conduct case studies with real deployments of these frameworks to validate the listed insider threats and to further develop mitigations.

References

1. Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N., Kastania, A.: ASTRAEA: a decentralized blockchain oracle. In: 2018 IEEE International Conference on Blockchain (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00207
2. Al Khalil, F., Butler, T., O'Brien, L., Ceci, M.: Trust in smart contracts is a process, as well. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 510–519. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_32
3. Amoroso, E., Nguyen, T., Weiss, J., Watson, J., Lapiska, P., Starr, T.: Toward an approach to measuring software trust. In: Proceedings of 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 198–218 (1991). https://doi.org/10.1109/RISP.1991.130788
4. Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, pp. 30:1–30:15. ACM, New York (2018). https://doi.org/10.1145/3190508.3190538
5. Aublin, P.L., Mokhtar, S.B., Quema, V.: RBFT: redundant byzantine fault tolerance. In: Proceedings of International Conference on Distributed Computing Systems, pp. 297–306 (2013). https://doi.org/10.1109/ICDCS.2013.53
6. Bansarkhani, R.E., Geihs, M., Buchmann, J.: PQChain: strategic design decisions for distributed ledger technologies against future threats. IEEE Secur. Priv. (2018). https://doi.org/10.1109/MSP.2018.3111246
7. Baracaldo, N., Joshi, J.: A Trust-and-risk aware RBAC framework: tackling insider threat. In: SACMAT 2012: Proceedings of the 17th ACM symposium on Access Control Models and Technologies (2012)
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Eprint.Iacr.Org (2018). https://doi.org/10.1016/j.bspc.2009.02.004
9. Bessani, A., Sousa, J., Alchieri, E.E.P.: State machine replication for the masses with BFT-SMART. In: DSN, vol. 6897, pp. 355–362, December 2014. https://doi.org/10.1109/DSN.2014.43

10. Cachin, C., Vukolic, M.: Blockchain consensus protocols in the wild. In: Richa, A.W. (ed.) 31st International Symposium on Distributed Computing (DISC 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 91, pp. 1:1–1:16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2017). https://doi.org/10.4230/LIPIcs.DISC.2017.1
11. del Castillo, M.: Blockchain 50: Billion Dollar Babies (2019). https://www.forbes.com/sites/michaeldelcastillo/2019/04/16/blockchain-50-billion-dollar-babies
12. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002). https://doi.org/10.1145/571637.571640
13. Chia, V., et al.: Rethinking blockchain security: position paper. In: 2018 IEEE International Conference on Blockchain (2018). http://arxiv.org/abs/1806.04358
14. Clement, A., Wong, E., Alvisi, L., Dahlin, M., Marchetti, M.: Making byzantine fault tolerant systems tolerate Byzantine faults. In: NSDI 2009: Proceedings of the 6th USENIX symposium on Networked systems design and implementation (2009). https://doi.org/10.1145/1989727.1989732
15. Collins, M., Cappelli, D.M., Caron, T., Trzeciak, R.F., Moore, A.P.: Spotlight on: programmers as malicious insiders-updated and revised. Technical report, Software Engineering Institute, Carnegie Mellon University (2013)
16. Colwill, C.: Human factors in information security: The insider threat - who can you trust these days? Information Security Technical Report (2009). https://doi.org/10.1016/j.istr.2010.04.004
17. Dasgupta, D., Shrein, J.M., Gupta, K.D.: A survey of blockchain from security perspective. J. Bank. Financ. Technol. (2019). https://doi.org/10.1007/s42786-018-00002-6
18. De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., Sassone, V.: PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain. CEUR Workshop Proceedings, vol. 2058, pp. 1–11 (2018)
19. Decan, A., Mens, T., Grosjean, P.: An empirical comparison of dependency network evolution in seven software packaging ecosystems. Empir. Softw. Eng. (2019). https://doi.org/10.1007/s10664-017-9589-y
20. Deventer, M.O., et al.: Techruption Consortium Blockchain: what it takes to run a blockchain together. In: Proceedings of 1st ERCIM Blockchain Workshop 2018, Amsterdam, Netherlands 8–9 May 2018. European Society for Socially Embedded Technologies (EUSSET) (2018)
21. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.L.: BLOCKBENCH: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD 2017, pp. 1085–1100. ACM, New York (2017). https://doi.org/10.1145/3035918.3064033. http://doi.acm.org/10.1145/3035918.3064033
22. Ekparinya, P., Gramoli, V., Jourjon, G.: The attack of the clones against proof-of-authority. CoRR (2019). http://arxiv.org/abs/1902.10244
23. ENISA: ENISA threat landscape report 2018. Technical report, ENISA (2019). https://doi.org/10.2824/622757
24. Ethereum Foundation: Go-Ethereum Website (2019). https://geth.ethereum.org
25. Franqueira, V.N.L., van Cleeff, A., van Eck, P., Wieringa, R.: External insider threat: a real security challenge in nterprise value webs. In: 2010 International Conference on Availability, Reliability and Security, pp. 446–453, February 2010. https://doi.org/10.1109/ARES.2010.40

26. Fröwis, M., Böhme, R.: In code we trust? In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 357–372. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_20
27. Fuchs, L., Pernul, G.: Minimizing insider misuse through secure Identity Management. Secur. Commun. Netw. (2012). https://doi.org/10.1002/sec.314
28. Gambetta, D.: Can we trust trust? In: Trust: Making and Breaking Cooperative Relations, pp. 213–237. Blackwell (1988)
29. Glaser, F.: Pervasive decentralisation of digital infrastructures: a framework for blockchain enabled system and use case analysis. In: HICSS 2017 Proceedings, pp. 1543–1552 (2017). https://doi.org/10.1145/1235
30. Hawlitschek, F., Notheisen, B., Teubner, T.: The limits of trust-free systems: a literature review on blockchain technology and trust in the sharing economy. Electron. Commer. Res. Appl. **29** (2018). https://doi.org/10.1016/j.elerap.2018.03.005
31. Hearn, M.: Corda: a distributed ledger (2016). https://docs.corda.net/head/_static/corda-technical-whitepaper.pdf
32. Hileman, G., Rauchs, M.: 2017 Global Blockchain Benchmarking Study (2017)
33. Huseby, D.: Security Code Audits - Hyperledger Wiki (2019). https://wiki.hyperledger.org/display/HYP/Security+Code+Audits
34. Hussain, S.R., Sallam, A., Bertino, E.: DetAnom: detecting anomalous database transactions by insiders. In: CODASPY 2015 - Proceedings of the 5th ACM Conference on Data and Application Security and Privacy (2015). https://doi.org/10.1145/2699026.2699111
35. Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q.: A survey on the security of blockchain systems. Futur. Gener. Comput. Syst. (2017). https://doi.org/10.1016/j.future.2017.08.020. http://www.sciencedirect.com/science/article/pii/S0167739X17318332
36. Litke, A., Anagnostopoulos, D., Varvarigou, T.: Blockchains for supply chain management: architectural elements and challenges towards a global scale deployment. Logistics **3**(1) (2019). https://doi.org/10.3390/logistics3010005
37. Loch, K.D., Carr, H.H., Warkentin, M.E.: Threats to information systems: today's reality, yesterday's understanding. MIS Q. (1992). https://doi.org/10.1163/18781527-00401005
38. Locher, T., Obermeier, S., Pignolet, Y.A.: When can a distributed ledger replace a trusted third party? In: IEEE International Conference on Blockchain (2018). http://arxiv.org/abs/1806.10929
39. Lustig, C., Nardi, B.: Algorithmic authority: the case of Bitcoin. In: Proceedings of the Annual Hawaii International Conference on System Sciences (2015). https://doi.org/10.1109/HICSS.2015.95
40. Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS 2016 (2016). https://doi.org/10.1145/2976749.2978309
41. Marti, S., Garcia-Molina, H.: Taxonomy of trust: categorizing P2P reputation systems. Comput. Netw. (2006). https://doi.org/10.1016/j.comnet.2005.07.011
42. MetaMask Contributors: MetaMask (2019). https://metamask.io/
43. Muskens, J., Chaudron, M.: Integrity management in component based systems. In: Proceedings of 30th Euromicro Conference, pp. 611–619 (2004). https://doi.org/10.1109/EURMIC.2004.1333429
44. R3: Corda Documentation (2019). https://docs.corda.net/releases/release-V4.1/

45. Sas, C., Khairuddin, I.E.: Exploring trust in Bitcoin technology: a framework for HCI research. In: Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction - OzCHI 2015 (2015). https://doi.org/10.1145/2838739.2838821
46. Schaffers, H.: The relevance of blockchain for collaborative networked organizations. In: Camarinha-Matos, L.M., Afsarmanesh, H., Rezgui, Y. (eds.) PRO-VE 2018. IAICT, vol. 534, pp. 3–17. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99127-6_1
47. Schneier, B.: Blockchain and Trust - Schneier on Security (2019). https://www.schneier.com/blog/archives/2019/02/blockchain_and_.html
48. Swihart, J., Winston, B., Bowe, S.: Zcash Counterfeiting Vulnerability Successfully Remediated - Zcash (2019). https://z.cash/blog/zcash-counterfeiting-vulnerability-successfully-remediated/
49. The Linux Foundation: Hyperledger Sawtooth Documentation (2019). https://sawtooth.hyperledger.org/docs/core/releases/1.1.5/contents.html
50. United States Department of Homeland Security: A Roadmap for Cybersecurity Research (2009). https://doi.org/10.1016/j.biortech.2007.06.061
51. Vo, H.T., Wang, Z., Karunamoorthy, D., Wagner, J., Abebe, E., Mohania, M.: Internet of blockchains: techniques and challenges ahead. In: 2018 IEEE International Conference on Blockchain. IEEE (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00264
52. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. In: Proceedings of 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, pp. 182–191. IEEE, April 2016. https://doi.org/10.1109/WICSA.2016.21
53. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: an authenticated data feed for smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016). https://doi.org/10.1145/2976749.2978326

2.2 Detecting Blockchain Security Threats [P6]

Status:	published
Publication:	2020 IEEE International Conference on Blockchain
Submitted:	15 June 2020
Accepted:	11 September 2020
Citation:	Benedikt Putz and Günther Pernul. 2020. Detecting Blockchain Security Threats. In 2020 IEEE International Conference on Blockchain (Blockchain), IEEE, 313–320.

Conference Description: The goal of the IEEE International Conference on Blockchain is to promote community-wide discussion for identifying advanced applications, technologies and theories for blockchain. It seeks submissions of papers that invent novel techniques, investigate new applications, introduce advanced methodologies, propose promising research directions and discuss approaches for unsolved issues.

2020 IEEE International Conference on Blockchain (Blockchain)

Detecting Blockchain Security Threats

Benedikt Putz, Günther Pernul
Chair of Information Systems
University of Regensburg
Regensburg, Germany
benedikt.putz@ur.de, guenther.pernul@ur.de

***Abstract*—In many organizations, permissioned blockchain networks are currently transitioning from a proof-of-concept stage to production use. A crucial part of this transition is ensuring awareness of potential threats to network operations. Due to the plethora of software components involved in distributed ledgers, threats may be difficult or impossible to detect without a structured monitoring approach. To this end, we conduct a survey of attacks on permissioned blockchains and develop a set of threat indicators. To gather these indicators, a data processing pipeline is proposed to aggregate log information from relevant blockchain components, enriched with data from external sources. To evaluate the feasibility of monitoring current blockchain frameworks, we determine relevant data sources in Hyperledger Fabric. Our results show that the required data is mostly available, but also highlight significant improvement potential with regard to threat intelligence, chaincode scanners and built-in metrics.**

***Index Terms*—distributed ledger, permissioned blockchain, information security, security monitoring, insider threat**

I. Introduction

Enterprise applications based on Distributed Ledger Technology (DLT) are no longer just proof-of-concept. They are now used in production environments to track shipping containers across the supply chain [1], settle trade finance deals [2] and to handle the exchange of valuable aerospace parts on a decentralized marketplace [3]. These use cases present attractive targets for internal and external attackers to exploit. While blockchains attempt to thwart attackers by replicating the database and code execution, there is still no shortage of attacks, as discovered by researchers and practitioners [4]. Attacks may target the consensus algorithm, flaws in smart contract programming languages, or flaws in the blockchain framework itself. Besides vulnerabilities, there are also operational security concerns such as private key or host system compromise. Security professionals looking to protect against these attacks need to have a clear idea what threats they are facing, how to detect ongoing attacks and how to protect against them. Commonly, Security Information and Event Management (SIEM) systems are used for such tasks, but currently no such SIEM system dedicated to permissioned blockchains exists. It is challenging for off-the-shelf SIEM systems to provide an integrated overview of a blockchain network's state, since there is a large number of components within a single blockchain node [5], whose behavior also depends on nodes outside the own organization's confines.

To illustrate the contribution of this work we use Jaquith's model of IT security controls [6]. While there is plenty of literature detailing threats [7] and exposures [4] of blockchains, approaches for countermeasures are still scarce. In the model shown in Figure 1, countermeasures consist of deterrent, detective, preventative and corrective controls. While there are some *deterrent* controls built into blockchains (such as the use of hashes and signatures for integrity preservation), *detective* controls to discover threats are still scarce.

To this end, we present a comprehensive study of attacks on permissioned blockchains. Based on our findings, we develop a set of threat indicators for automated attack detection based on log data (detective control). We focus specifically on the permissioned blockchain framework Hyperledger Fabric, which is used by almost half of the world's biggest companies evaluating DLT (23 out of 50 companies with more than $1 billion valuation) [8].

In summary, we contribute to research by

- providing a **comprehensive overview of possible attacks** on permissioned blockchains
- developing a **set of blockchain threat indicators** for attacks on permissioned blockchains
- investigating the **feasibility of monitoring attacks on Hyperledger Fabric** and identifying areas for future research and development

The remainder of this paper is structured as follows. We first provide an overview of related work in Section II, before defining the threat model for monitoring in Section III. In Section IV we present a study of possible attacks on permissioned blockchain networks, and define threat indicators for each attack. A suitable data processing architecture to gather

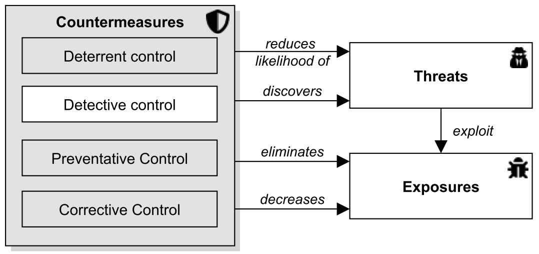


Fig. 1. Logical model of IT Security Controls [6], with the missing DLT detective control highlighted

978-0-7381-0495-9/20/$31.00 ©2020 IEEE
DOI 10.1109/Blockchain50366.2020.00046

these indicators from blockchain data sources is defined in Section V. We evaluate the developed monitoring approach by investigating the feasibility of indicator collection with Hyperledger Fabric in Section VI. Finally, we discuss SIEM and organizational integration of blockchain monitoring in Section VII.

II. Related Work

Currently, the majority of proposed monitoring tools and methods focus on isolated detection of anomalies within specific components of a DLT node. BAD [9] is designed to detect anomalous Bitcoin transactions to prevent transactions with malicious payloads from spreading in the network. LedgerGuard [10] monitors a Hyperledger Fabric peer's ledger and restores corrupted blocks from connected peers if needed. Garcia et al. propose Lazarus [11], a solution for diversity management of consensus nodes. Lazarus monitors the software stack of each blockchain node for newly found or zero-day exploits and quarantines affected replicas. A number of tools exist for formal verification and vulnerability detection of Ethereum smart contracts, including both offline scanners [12], [13] and online detection frameworks [14].

There are also several tools focusing on performance and availability monitoring. Hyperledger Caliper [15] focuses on performance benchmarking in an isolated testing environment. Hyperledger Explorer [16] displays basic information about running nodes, allows users to inspect blockchain state and interact with deployed chaincode. However, its main purpose is browsing activity on the underlying blockchain network, not monitoring the network for security threats.

Compared to these approaches, the present work provides a holistic perspective on blockchain security monitoring, instead of only focusing on isolated aspects. Based on a comprehensive survey of attacks on permissioned distributed ledgers, we determine suitable threat indicators and corresponding data sources to compute them. Going beyond plain status monitoring offered by tools like Hyperledger Explorer, we focus on extracting security-relevant threat indicators from Hyperledger Fabric.

III. Threat Model

Before going into detail on threat indicators, a threat model of possible attackers and attacks needs to be established. In this Section we provide a brief overview of DLT actors and DLT-specific threats (split into vulnerabilities and malicious intent), followed by a detailed enumeration of attacks in Section IV.

A. Actors

As a first step, we model the actors in a blockchain network. Figure 2 shows a data flow context diagram containing relevant actors. Each actor may cause a threat, either by acting as a malicious insider or as an external attacker. *Transactors* are regular users, which have read access to the blockchain and can submit transactions. *Peer*, *Orderer* and *Certificate Authority (CA) Admins* administrate the corresponding blockchain components and have special privileges. *External Users* have no privileges and are locked out of the system by access control, barring vulnerabilities. Figure 3 shows a Level 1 data flow diagram detailing how these actors interact with the processing nodes within the blockchain system (modeled after Hyperledger Fabric). Each *Admin* controls the corresponding blockchain component. Outside access is provided to *Transactors* and *External Users* via an API.

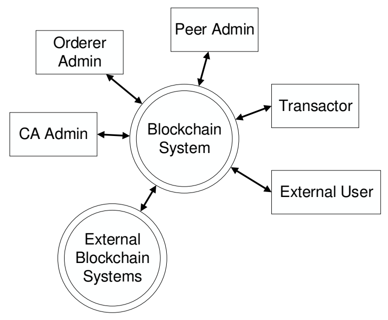


Fig. 2. Level 0 data flow diagram of blockchain actors

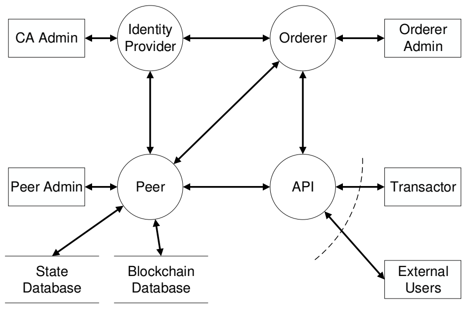


Fig. 3. Level 1 data flow diagram of blockchain actors

B. Vulnerabilities

Vulnerabilities increase the exposure to threats by providing attackers with ways to compromise the protection goals confidentiality, integrity and availability. Like any software, distributed ledger frameworks are prone to software bugs, which may result in vulnerabilities. A prime example are **vulnerable protocols**, caused by implementation bugs in cryptographic, networking or storage components or dependencies. For example, Hyperledger Fabric uses gRPC for exchanging blocks, which has been subject to a number of high and critical severity Common Vulnerability Enumerations (CVEs)[1]. Another type of exposure are **vulnerable contracts**, where the intricacies of smart contract development can lead to exploitable behavior [13].

Distributed ledger frameworks also offer a large number of configuration options. Since these options are rarely doc-

[1]see https://nvd.nist.gov/

umented in a single place, negligence or oversight may lead to **misconfiguration** by administrators. Bad configuration and a lack of consideration for security on deployment of blockchain nodes then increases exposure to attacks.

C. Malicious intent

Internal threats. Distributed Ledgers are subject to a variety of threats by insiders and external insiders [17], who may attempt to exploit the blockchain for personal or organizational gain. Besides inadvertent misconfigurations, insiders may intentionally manipulate the configuration of blockchain peers and threaten network security. Through initiation of updates for smart contracts they may introduce vulnerabilities or backdoors. For these reasons, administrators are potential single points of failure for an organization's blockchain node and their actions should be monitored by an independent information security team.

External attackers. External adversaries may attempt to gain blockchain network access in order to read and possibly manipulate ledger data. To this end, attackers may exploit the aforementioned vulnerabilities. A successful attack could result in subversion of a blockchain peer or identity provider, which is a prerequisite for many attacks only possible from inside the network. Denial of service attacks are an additional attack vector, which may stall consensus if sufficient nodes are affected.

IV. Attacks and Threat Indicators

To gain an overview of relevant attacks, we conducted a literature review of attacks on permissioned blockchain systems. We searched for the terms `("permissioned blockchain" OR "hyperledger fabric") AND ("vulnerability" OR "attack")`, using the ACM digital library (106 results), SpringerLink (361 results), ScienceDirect (218 results), IEEE Xplore (10 results) and the Wiley Online Library (55 results). We filtered these results for works dealing with attacks and vulnerabilities on permissioned blockchains, which left us with 10 papers. The low number of filtered results can be attributed to the fact that many papers mention or cite existing vulnerabilities and attacks, but do not contribute new vulnerabilities. We conducted additional in-depth research on each of the found attacks by searching for the attack's name, which yielded additional literature [4], [7], [18].

We extract all attacks applicable to permissioned blockchains from the surveyed papers. For conciseness, some attacks are grouped under a common term (i.e. Contract Vulnerabilities). Based on the identified attacks and affected blockchain components, we develop threat indicators that allow a security expert to recognize ongoing attacks. While we focus on applicability of the attacks on Hyperledger Fabric, many of the attacks are applicable to other permissioned blockchain frameworks as well. The categorized overview of attacks is shown in Table I. Generally, threat indicators may be *proactive* or *reactive*. Proactive approaches attempt to detect the vulnerability before exploitation, while reactive approaches detect the act of exploitation and attempt to limit the damage.

For the remainder of this chapter, we go into more detail on each attack and how it can be detected with suitable indicators. Based on the threat model, we first focus on *vulnerabilities* followed by attacks of *malicious intent*. These indicators are then elaborated in Section VI with regard to Hyperledger Fabric.

A. Vulnerabilities

Contract Vulnerability. A contract vulnerability refers to a security bug in a smart contract that must be fixed through a contract upgrade. Since an organization may not have control over all contracts that it is sending transactions to, it is important to also monitor contracts owned by other organizations on the network. In general, contract vulnerabilities are difficult to detect, since abuse transaction patterns vary depending on the vulnerability. For example, the *Re-entrancy attack* on Solidity smart contracts [19] may be detected by excessive resource consumption from a single transaction. Other vulnerabilities may require inspection of contract state changes, such as the *delegatecall injection* [13]. Yamashita et al. provide an overview of Hyperledger Fabric chaincode risks related to non-determinism, phantom state database reads and unchecked inputs [24]. Unchecked inputs may for example result in JSON injection vulnerabilities [25]. By scanning each contract deploy/upgrade transaction, new vulnerabilities can be derived from in the scan logs. The number of *scanned potential vulnerabilities* in deployed contracts is a useful indicator for monitoring and reducing the attack surface of smart contracts.

Framework Vulnerability. The code of the blockchain framework may be subject to vulnerabilities. This category includes vulnerabilities such as *insufficient smart contract virtualization* [20], [26] and *injection of malicious code* due to improper input checking [26]. Correspondingly, *framework releases* should be monitored for such vulnerabilities to upgrade to new versions as soon as possible.

Dependency Vulnerability. Blockchain frameworks also rely on a number of *direct* and *transitive* dependencies. A major category of dependencies are database systems used for storage of blockchain state. Most blockchain frameworks use self-sufficient DBMS such as LevelDB, CouchDB or Postgres [17]. As dependencies, their versions are often updated infrequently and insufficient default configurations are used. For example, the CouchDB instance preconfigured with Hyperledger Fabric was found to be susceptible to direct unauthenticated manipulation by via the built-in web interface, voiding integrity assumptions of the framework [20]. This vulnerable default configuration is also present in the latest test-network provided with Fabric 2.1. While this should be prevented by securely configuring the dependency initially, monitoring *configuration changes* and *database container logs* for foreign IP access would also detect exploitation.

Cryptographic Vulnerability. Vulnerabilities in cryptographic protocols are a serious threat, since blockchain frameworks rely on hashes and digital signatures for integrity,

TABLE I
OVERVIEW OF ATTACKS AND CORRESPONDING THREAT INDICATORS (BOTH VULNERABILITIES AND MALICIOUS INTENT).

Attack Category	Attack Examples	Threat Indicators	Type
Contract Vulnerability	Reentrancy [19], delegatecall [13], Dependency injection [17]	scanned potential vulnerabilities threat intelligence on vulnerabilities	Proactive Proactive
Framework Vulnerability	Unrestricted Chaincode Containers [20]	framework releases	Proactive
Dependency Vulnerability	CouchDB web interface [20]	threat intelligence on vulnerabilities dependency container logs	Proactive Reactive
Cryptographic Vulnerability	Quantum Computing Threat [17], Hash Collision Resistance Attack [17]	threat intelligence on vulnerabilities	Proactive
Denial of Service	Dust transactions [4], Storage pollution [20]	transaction throughput transaction latency incoming network messages oustanding transactions	Reactive
Network Partitioning	BGP hijacking [21], DNS attacks [4], [22], Eclipse attack [4], [22], Attack of the Clones [21]	connected peers	Reactive
Malicious Consensus Behavior	Consensus Delay [4], Alternative History [7], [20], Block Withholding [4], Transaction Reordering [20]	discarded blocks latest block hashes leader election frequency outstanding transactions (age) client application outgoing transactions	Reactive
Consensus Configuration Manipulation	Batch Time attack [20], Block Size attack [20]	configuration changes configuration value bounds	Proactive
Identity Provider Compromise	CA Attack [20], [22], Sybil attacks [18], [20], [23], Boycott attack [20], Blacklisting attack [20]	certificate requests (successful/denied) certificate revocations transactor identities	Reactive

authentication and non-repudiation. If SHA256 were to be affected by a collision-resistance attack similar to the one discovered for SHA-1 (CVE 2005-4900), most major blockchain frameworks would be affected [17]. Detecting such an attack is only possible by *monitoring threat intelligence*, i.e. new CVEs in the NIST database.

B. Malicious Intent

Denial of Service (DoS). While blockchain systems are less threatened by DoS than centralized servers due to built-in replication and fault-tolerance, targeted attacks still represent a threat. To achieve DoS, an outsider may attempt to flood specific or multiple blockchain peers with TCP syn packets. In particular, DoS attacks targeting the consensus leader can significantly reduce or stall consensus entirely [17]. Internal attackers with access to the network can simply send a large number of transactions (or transactions with a large size), which must all be processed by endorsing peers [4]. Even if they are invalid they are included in the blockchain by all peers, polluting the storage [20]. Consequently, indicators for DoS attacks are low *transaction throughput* and high *transaction latency* [27]. Since these metrics are also affected by other factors (such as network usage by regular Transactors), monitoring *incoming network messages* and *incoming transactions* provides a more comprehensive picture.

Network Partitioning. Internal attackers may attempt to partition a blockchain network by manipulating network routing [4]. The goal of such an attack is manipulation of the consensus protocol. This can be accomplished through network-level attacks such as *BGP hijacking* [21] and *DNS attacks* [4], [22]. This attack can then be followed up with consensus manipulation attacks. Examples of attacks based on network partitioning include the *Attack of the Clones* on the Proof-of-Authority consensus protocols [21][2] and *Eclipse attacks* [4]. In permissioned networks, monitoring the number of *connected peers* can be used to detect network partitioning attempts.

Malicious Consensus Behavior. Orderer Admins can launch a variety of attacks by behaving maliciously during consensus. A *Consensus Delay* attack can be initiated by peers propagating invalid blocks [4]. The *Intentional Fork* attack in Hyperledger Fabric describes a similar scenario where the ordering service sends out conflicting versions of blocks to peers [20]. Both attacks result in consensus delay since the blockchain peers waste computing power on verifying invalid blocks. They can be detected by monitoring the number of *discarded blocks* that were received through peer-to-peer communication.

If several Orderer Admins collude, they may attempt to rewrite the blockchain in an *Alternative History* attack [7], [20]. This requires $> 50\%$ of nodes to collude (crash-fault tolerance), or $> 2f$ nodes for byzantine-fault tolerant consensus where $3f + 1$ nodes tolerate f malicious nodes. To detect any state forks or attempts at rewriting history, the *latest block hashes* of all peers must be monitored and compared.

Byzantine attacks become possible when a non byzantine-fault tolerant (BFT) consensus algorithm is used. Hyperledger Fabric 2.1 only offers Kafka and Raft implementations, which

[2]applicable to permissioned networks based on Ethereum

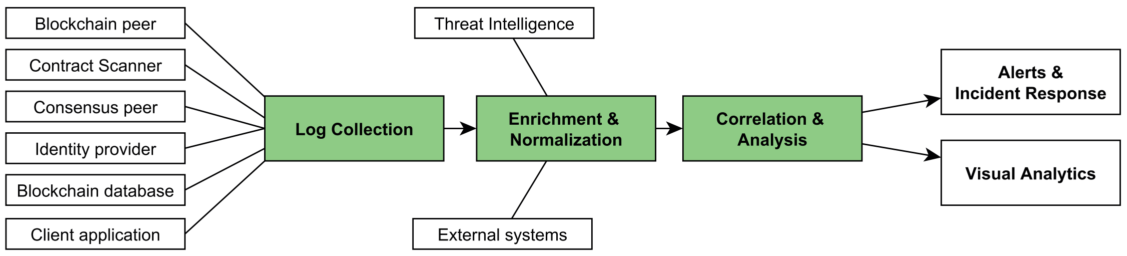


Fig. 4. Proposed Blockchain Security Monitoring pipeline.

are merely crash-fault tolerant. A malicious Raft node may prevent consensus indefinitely by constantly starting new leader elections, or cause correctness violations if elected as leader [28]. Leader election misbehavior can be detected by monitoring *leader election frequency*.

A malicious leader (also possible in BFT consensus [4]) is more difficult to detect, since it may cause different types of correctness violations. For example, during a *Block Withholding* [4] or Sabotage [20] attack the consensus leader or ordering service witholds blocks containing unwanted transactions, or transactions from specific participants. This attack can be detected by monitoring the *age of outstanding transactions* in the transaction pool. Transactions with a large age indicate that the orderer cluster is not reaching consensus on them.

Another example of a correctness violation is a *Transaction Reordering* attack, where the leader of the ordering service reorders transactions to favor specific organizations [20]. Orderer Admins might abuse this to gain an advantage in smart contracts where timing is critical. If an organization relies on such timing-critical contracts, it should track *client application outgoing transactions*. When the transaction is eventually included in a block, reordering can be detected by comparing timestamps.

Consensus Configuration Manipulation. If an attacker controls a majority of consensus nodes, it becomes possible to manipulate the consensus process by changing configuration values. For the Hyperledger Fabric Ordering service, the *Batch Time attack* and *Block Size attack* are known [20]. Both can delay transactions indefinitely by increasing the time until transactions are included in a block. To mitigate this threat, *configuration changes* should be monitored to detect unsafe configuration values outside of specified bounds before they are approved by Peer Admins.

Identity Provider Compromise. In Hyperledger Fabric, identity providers are referred to as Membership Service Providers (MSPs) and the default implementation is called Fabric-CA. A Fabric-CA MSP may be compromised through private key theft, also referred to as a *CA Attack* [22]. The actual theft of private keys cannot be detected from the blockchain framework's perspective, but malicious actions using these keys can be discovered. A frequently-cited example are *sybil attacks*, where a single attacker forges multiple identities [18], [23], i.e. with certificates from a compromised MSP. These identities may be used to circumvent contract endorsement policies, and thus manipulate contract execution [20]. Sybil attacks can be detected by closely monitoring *newly issued certificates*. Since this may not be possible for certificates issued by external MSPs, *Transactor identities* should also be monitored.

The *Boycott attack* refers to a scenario where two organizations are under the same MSP and one of them is denied new certificates [20]. Consequently, the CA should be monitored for *denied certificate requests*. The *Blacklisting Attack* is based on revoked certificates and may result in peers or Transactors losing network access [20], [23]. Since certificate revocations should normally occur rarely, the *number of revoked certificates* is of interest.

V. Data Collection and Processing

To assemble the threat indicators developed above, a pipeline for data collection and processing is needed. The goal of data processing is to provide an aggregated view of the threat indicators, enabling an expert to detect security threats.

We derive the data processing pipeline from the SIEM pattern [29]. Initially, data is collected from internal blockchain data sources (*log collection*) and enriched with external data. Subsequently it must be normalized to a common data format (*enrichment & normalization*). The following steps are outside the scope of this paper, but briefly discussed in Section VII. Correlating multiple indicators (*Correlation & Analysis*), visualizing them (*Visual Analytics*) and triggering alerts (*Alerts & Incident Response*) provide valuable aid to security experts.

Log Collection. Logs are collected from various sources producing different types of information: numeric metrics (i.e. transactions per second), application log events and blockchain state events. Depending on the framework they are retrieved via *push* or *pull* mechanisms. A summary of data sources for Hyperledger Fabric is shown in Table II. The Operations Service provides numeric metrics about current operation, which can be consumed by a Prometheus (pull) or a StatsD instance (push). Peer channel-based event services allow an agent to subscribe to channel-specific block data (push), which also includes application-level chaincode events. Log data is provided by the Docker containers of the Hyperledger Fabric components, which provide logs of configurable log level detail (such as `INFO` and `DEBUG`). This includes auxiliary services

TABLE II
HYPERLEDGER FABRIC 2.1 DATA SOURCES AND METRICS [30].

Data	Type	Source	Collection
Numeric Metrics	Numeric	Prometheus (Peer, Orderer, MSP)	Pull
Numeric Metrics	Numeric	StatsD (Peer, Orderer, MSP)	Push
Channel Events	Application data	SDK (Peer)	Push
Logs	Behavioral	Docker (Peer, Chaincode, Orderer, MSP, CouchDB)	Pull
Blockchain state/history	Application data	SDK (Peer)	Pull

such as Fabric-CA (identity provider) and CouchDB (state database provider). Finally, the full spectrum of blockchain data is available via the Peer SDK, but data must be queried on demand (pull). In addition to these built-in data sources, chaincode vulnerability scanners provide log files for ingestion.

Enrichment & Normalization. Events occurring within the organization are enriched with contextual data from external systems. If accessible, data from connected blockchain peers' APIs should be collected. Such external peer data can help determine whether an incident is isolated or network-wide.

Depending on the blockchain setup, additional data sources may be needed. There are several optional blockchain features that involve external systems:

- **Permissionless Blockchain Anchoring**: The anchoring status needs to be monitored in case the anchoring chain is broken. Periodic API requests to a node on the anchoring target blockchain can provide the required data.
- **Oracles**: Oracles provide external data to smart contracts. If an oracle is manipulated or compromised the consequences can be severe for the relying contract. Thus, data provided by oracles should be monitored for anomalies.
- **Cross-chain interactions**: Cross-chain interactions such as hash-timelocked contracts are used to exchange assets between blockchains. The status of these asset swaps on blockchains other than the primary monitoring target should be tracked, in case there is an issue with contracts on either side of the swap.
- **Off-chain storage**: Blockchain applications often link data on the blockchain via hashes for timestamping and non-repudiation purposes. The availability of the linked data should be periodically checked.

VI. EVALUATION

To validate the detectability of the indicators outlined above in Table I, we conducted experiments with a Hyperledger Fabric deployment. We inspected log files and data sources hands-on to determine whether attacks can actually be detected with currently provided data sources. Hereafter, we elaborate for each attack which data source is suitable, and how these data sources could be improved to enhance detectability.

Scanned potential vulnerabilities. To our knowledge, only two vulnerability scanners exist for Hyperledger Fabric, both closed-source and only supporting Go chaincode (Chaincode Scanner[3] and an unnamed tool [24]). The coverage of these tools can be augmented by language-specific ones such as gosec[4]. However, there is a clear need for open-source chaincode scanners with support for all chaincode languages.

Threat intelligence on vulnerabilities. To be able to detect new vulnerabilities and upgrade network nodes as soon as possible, threat intelligence feeds need to be monitored constantly for vulnerabilities relevant to blockchain components. However, searching the NIST National Vulnerability database for "Hyperledger Fabric" yields 0 results. As of today, there are no blockchain-specific sources of threat intelligence information. Nevertheless, threat intelligence feeds can be monitored for relevant keywords (i.e. *gRPC*, *CouchDB*, *Golang*, *SHA256*) and manually filtered for applicable threats. Additional sources can be community collaboration tools for open-source frameworks (i.e. monitoring the Hyperledger JIRA[5] or mailing list for the keyword *security*).

Framework releases. Hyperledger Fabric provides a GitHub feed with the latest releases[6]. Each release has sections on known and resolved vulnerabilities, which can be used to determine if a timely upgrade is needed.

Dependency Container Logs. If CouchDB is used with Hyperledger Fabric, all requests are logged to the Docker container logs including the IP address, time and request URL. This information can be used to detect suspicious requests from foreign IP addresses and other types of attacks.

Transaction throughput. Throughput can be observed based on Fabric's ordering service metric `broadcast_processed_count`, which counts the number of processed transactions over time. Alternatively, this metric can be computed based on the transactions contained in block data, which is needed for other metrics (i.e. latest block hashes).

Transaction latency. Transaction latency can be computed as the delay between transaction timestamp and the block timestamp of the transaction's block. This metric is thus computed for each transaction upon block inclusion. By subscribing to Hyperledger Fabric's Channel Events, this metric can be computed for each transaction in each block signed by the ordering service.

Incoming network messages. The operations services provides numerous metrics for monitoring network communication. `gossip_comm_messages_received` tracks messages received via peer gossip, and the `grpc_server_*` metrics track gRPC communication with clients.

[3]https://chaincode.chainsecurity.com/

[4]https://github.com/securego/gosec

[5]https://jira.hyperledger.org

[6]https://github.com/hyperledger/fabric/releases

Outstanding transactions. To our knowledge, there is no metric that keeps tracks of outstanding unprocessed transactions. This would be beneficial to determine the cause of unprocessed transactions (high system load or deliberate exclusion). For transaction delay attacks, the operations service provides the histogram metric `blockcutter_block_fill_duration`, which monitors the time from transaction enqueuing at the orderer to inclusion in a block. By monitoring transactions fill duration over time, the average age of outstanding transactions can be determined.

Connected peers. Ordering service outgoing connections can be measured using the metric `cluster_comm_egress_tls_connection_count`, while peer connections are evident from `grpc_comm_conn_opened`. While isolated disconnections may also be caused by a benign crash fault, multiple can indicate a network partition.

Discarded blocks. The peer container logs provide information about received and validated blocks, which can be used to determine which blocks were successfully validated. Since this requires preprocessing, a built-in metric would be desirable.

Leader election frequency. For Hyperledger Fabric's Raft consensus, the metric `consensus_etcdraft_leader_changes` provides a counter for leader changes, and orderer log files indicate which node was elected. Tracking leader changes over time can provide an indication of potential attacks. It would be beneficial if the consensus algorithm also indicated failed leader elections, to be able to detect nodes continuously proposing elections.

Latest block hashes. Block hashes can be computed based on the block headers, which are obtained through channel block event subscription. A block hash in Hyperledger Fabric is the SHA256 hash of the block number, the block data hash and the previous block's hash. Tracking the correctness of block hashes over time provides certainty that no blockchain reorganization has occurred.

Client application outgoing transactions. Outgoing transactions signed by a Hyperledger Fabric SDK contain the transaction hash and timestamp. These can be sent to and tracked by a monitoring service (push).

Configuration changes. Since timely notification about configuration updates is essential, this metric should also be derived through block subscription to the Channel Event Hub. Configuration transactions are identified by the type `CONFIG_UPDATE`. Additionally, the operations service provides the metric `consensus_etcdraft_config_proposals_received`, which keeps tracks of configuration transaction proposals.

Certificate requests and revocations. When using Fabric-CA server, certificate creation can be monitored by counting POST requests to `/enroll` (`[INFO]` log level). Successful requests have status code 2xx, while denied requests are evident from status codes 4xx and 5xx. For details such as the name of the registered identity, `[DEBUG]` log level is required. Revocations are monitored using log entries for requests to the `/revoke` endpoint.

Transactor identities. The node.JS SDK provides a way to subscribe to new blocks of a channel (`registerBlockEvent` method). The listener receives all transactions as part of the block, and each transaction contains the full sender certificate as part of the signature. Using these certificates, a monitoring system can keep track of existing identities.

VII. Discussion

After demonstrating practical feasibility of our concept, we now take a broader perspective and focus on integration with existing security monitoring facilities. This includes SIEM systems, but also Security Operations Centers (SOCs), which interpret data provided by monitoring systems.

A. SIEM integration

The monitoring architecture shown in Figure 4 can be implemented by connecting the data sources in Table II to SIEM software. In the next step, the available data needs to be normalized to a common format. Based on normalized data, correlation rules can be derived that pinpoint the source attack of the issue with certainty. If sufficient data is available to indicate an ongoing attack, security alerts can be triggered to prompt a detailed investigation. Thereto there is also significant potential to help experts interpret blockchain data and metrics by providing appropriate visualizations, for example as part of a monitoring dashboard.

B. Organizational aspects

Figure 5 shows how a Blockchain Security Monitor (shortened *BCSM*) interacts with human and software components. The security operations team monitors activities on the blockchain network and identifies the origin of potential threats. The *BCSM* only monitors blockchain components running on systems controlled by its operating organization. Data from nodes controlled by other organizations in the consortium may contribute to a complete overview of the threat situation, but cannot be relied upon.

To be able to detect manipulation by blockchain administrators, the security operations team must be separate from the blockchain operations team. It assumes a watchdog role that can help prevent insider attacks. *BCSM* is the technical

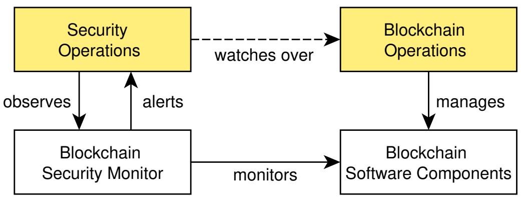


Fig. 5. Organizational integration of blockchain security monitoring.

component that supports this organizational function with informationa and alerts.

VIII. Conclusion

In this paper, we presented a review of attacks on permissioned blockchains. Based on the attacks, we developed indicators for an individual organization in a blockchain network to detect ongoing attacks, either proactively or reactively depending on the attack. For these metrics, we proposed a suitable data processing architecture inspired by SIEM systems. The feasibility of this architecture was demonstrated by closely examining the data sources that Hyperledger Fabric offers.

We also identified several areas of improvement for blockchain security. For example, adding metrics such as discarded blocks and failed leader elections to Hyperledger Fabric facilitates security monitoring. Additionally, open-source threat scanners for Hyperledger Fabric chaincode and blockchain-specific threat intelligence feeds can help patch vulnerabilities before they can be exploited. Finally, potential for malicious consensus behavior could be significantly reduced by introducing an ordering service based on BFT consensus for Hyperledger Fabric.

References

[1] T. Jensen, J. Hedman, and S. Henningsson, "How TradeLens delivers business value with blockchain technology," *MIS Quarterly Executive*, 2019.

[2] IBM, "we.trade | IBM," 2020. [Online]. Available: https://www.ibm.com/case-studies/wetrade-blockchain-fintech-trade-finance

[3] The Linux Foundation, "Honeywell Case Study – Hyperledger," 2020. [Online]. Available: https://www.hyperledger.org/resources/publications/honeywell-case-study

[4] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and A. Mohaisen, "Exploring the Attack Surface of Blockchain: A Systematic Overview," 2019.

[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: ACM, 2018, pp. 30:1–30:15.

[6] A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley Professional, 2007.

[7] I. Homoliak, S. Venugopalan, Q. Hum, D. Reijsbergen, R. Schumi, and P. Szalachowski, "The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses," 2019.

[8] M. del Castillo and M. Schifrin, "Blockchain 50," 2020. [Online]. Available: https://www.forbes.com/sites/michaeldelcastillo/2020/02/19/blockchain-50

[9] M. Signorini, M. Pontecorvi, W. Kanoun, and R. D. Pietro, "BAD: Blockchain Anomaly Detection," *CoRR*, vol. abs/1807.0, 2018. [Online]. Available: http://arxiv.org/abs/1807.03833

[10] Q. Zhang, P. Novotny, S. Baset, D. Dillenberger, A. Barger, and Y. Manevich, "LedgerGuard: Improving Blockchain Ledger Dependability," pp. 1–8, 2018. [Online]. Available: http://arxiv.org/abs/1805.01081

[11] M. Garcia, A. Bessani, and N. Neves, "Lazarus: Automatic Management of Diversity in BFT Systems," in *Proceedings of the 20th International Middleware Conference*, ser. Middleware '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 241–254. [Online]. Available: https://doi.org/10.1145/3361525.3361550

[12] M. D. Angelo and G. Salzer, "A Survey of Tools for Analyzing Ethereum Smart Contracts," in *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPCON 2019, Newark, CA, USA, April 4-9, 2019*. IEEE, 2019, pp. 69–78. [Online]. Available: https://doi.org/10.1109/DAPPCON.2019.00018

[13] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks and Defenses," *CoRR*, vol. abs/1908.0, 2019. [Online]. Available: http://arxiv.org/abs/1908.04507

[14] T. Chen, R. Cao, T. Li, X. Luo, G. Gu, Y. Zhang, Z. Liao, H. Zhu, G. Chen, Z. He, Y. Tang, X. Lin, and X. Zhang, "SODA: A Generic Online Detection Framework for Smart Contracts," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020*. The Internet Society, 2020.

[15] The Linux Foundation, "Hyperledger Caliper," 2020. [Online]. Available: https://www.hyperledger.org/use/caliper

[16] ——, "Hyperledger Explorer," 2020. [Online]. Available: https://www.hyperledger.org/use/explorer

[17] B. Putz and G. Pernul, "Trust Factors and Insider Threats in Permissioned Distributed Ledgers," *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, vol. XLII, pp. 25–50, 2019. [Online]. Available: http://link.springer.com/10.1007/978-3-662-60531-8_2

[18] D. Dasgupta, J. M. Shrein, and K. D. Gupta, "A survey of blockchain from security perspective," *Journal of Banking and Financial Technology*, 2019.

[19] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," in *Lecture Notes in Computer Science*, 2017.

[20] A. Dabholkar and V. Saraswat, "Ripping the Fabric: Attacks and Mitigations on Hyperledger Fabric," in *Applications and Techniques in Information Security*, V. S. Shankar Sriram, V. Subramaniyaswamy, N. Sasikaladevi, L. Zhang, L. Batten, and G. Li, Eds. Singapore: Springer Singapore, 2019, pp. 300–311.

[21] P. Ekparinya, V. Gramoli, and G. Jourjon, "The Attack of the Clones Against Proof-of-Authority," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020*. The Internet Society, 2020.

[22] A. Davenport, S. Shetty, and X. Liang, "Attack Surface Analysis of Permissioned Blockchain Platforms for Smart Cities," in *2018 IEEE International Smart Cities Conference, ISC2 2018*, 2019.

[23] J. Holbrook, "Blockchain Security and Threat Landscape," in *Architecting Enterprise Blockchain Solutions*. Wiley, 2020, pp. 323–347. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119557722.ch11

[24] K. Yamashita, Y. Nomura, E. Zhou, B. Pi, and S. Jun, "Potential Risks of Hyperledger Fabric Smart Contracts," in *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 2019, pp. 1–10.

[25] S. Riedesel, P. Hakimian, K. Buyens, and T. Biehn, "Tineola: taking a bite out of enterprise blockchain," 2018. [Online]. Available: https://github.com/tineola/tineola

[26] G. Shaw, "Hyperledger Fabric Security Audit," Nettitude, Tech. Rep., 2017. [Online]. Available: https://wiki.hyperledger.org/display/fabric/Audits?preview=/2393550/2393584/technical_report_linux_foundation_fabric_august_2017_v1.1.pdf

[27] N. Andola, Raghav, M. Gogoi, S. Venkatesan, and S. Verma, "Vulnerabilities on Hyperledger Fabric," *Pervasive and Mobile Computing*, 2019.

[28] C. Copeland and H. Zhong, "Tangaroa: a Byzantine Fault Tolerant Raft," 2016.

[29] M. Vielberth and G. Pernul, "A Security Information and Event Management Pattern," in *12th Latin American Conference on Pattern Languages of Programs (SLPLoP)*, nov 2018. [Online]. Available: https://epub.uni-regensburg.de/41139/

[30] The Linux Foundation, "Hyperledger Fabric 2.1 Documentation," 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.1

2.3 HyperSec: A Visual Analytics approach to blockchain monitoring [P7]

Status:	published
Publication:	ICT Systems Security and Privacy Protection - IFIP SEC 2021
Submitted:	21 January 2021
Accepted:	22 March 2021
Citation:	Benedikt Putz, Fabian Böhm, and Günther Pernul. 2021. HyperSec: Visual analytics for blockchain security monitoring. In ICT systems security and privacy protection - 36th IFIP TC 11 International Conference, SEC 2021, Oslo, Norway, June 22-24, 2021, Proceedings (IFIP Advances in Information and Communication Technology), Springer, 165–180.

Conference Description: The SEC conferences are a series of well-established international conferences on Security and Privacy. SEC is the flagship event of the International Federation for Information Processing (IFIP) Technical Committee 11 (TC-11). The IFIP SEC conferences aim to bring together primarily researchers, but also practitioners from academia, industry and governmental institutions to elaborate and discuss IT Security and Privacy Challenges that we are facing today and will be facing in the future.

HyperSec: Visual Analytics for Blockchain Security Monitoring

Benedikt Putz(✉), Fabian Böhm, and Günther Pernul

University of Regensburg, Regensburg, Germany
{benedikt.putz,fabian.boehm,guenther.pernul}@ur.de

Abstract. Today, permissioned blockchains are being adopted by large organizations for business critical operations. Consequently, they are subject to attacks by malicious actors. Researchers have discovered and enumerated a number of attacks that could threaten availability, integrity and confidentiality of blockchain data. However, currently it remains difficult to detect these attacks. We argue that security experts need appropriate visualizations to assist them in detecting attacks on blockchain networks. To achieve this, we develop HyperSec, a visual analytics monitoring tool that provides relevant information at a glance to detect ongoing attacks on Hyperledger Fabric. For evaluation, we connect the HyperSec prototype to a Hyperledger Fabric test network. The results show that common attacks on Fabric can be detected by a security expert using HyperSec's visualizations.

Keywords: Distributed ledger · Permissioned blockchain · Information security · Visual analytics · Security monitoring

1 Introduction

New use cases of distributed ledger technology (DLT) are proposed on a daily basis by academia and practice, leading to an increasing number of projects and solutions. Beyond that, blockchain applications are increasingly being used in real large-scale supply chain environments, such as the TradeLens [10] and DLFreight [20] platforms. At first glance, DLT seems to increase an application's security or even solve existing applications' security issues. However, the task of securing the DLT itself is often neglected in practice due to its complexity and the number of serious challenges connected to it.

The complexity of blockchain technology makes it particularly challenging to identify malicious activities [4]. In any blockchain network, there are several independent peers operated by independent organizations, where each organization only has a limited view of the network. Each node also has various data sources from its components, making it difficult to obtain an overview of the

B. Putz and F. Böhm—Contributed equally to this manuscript.

A. Jøsang et al. (Eds.): SEC 2021, IFIP AICT 625, pp. 165–180, 2021.
https://doi.org/10.1007/978-3-030-78120-0_11

network's state [17]. Since blockchain is a networked database, it also requires monitoring both the host and the network, which results in a large volume and velocity of observable data.

Fully automated systems for live attack detection on blockchains do not yet exist. Even if respective technologies for blockchain security monitoring were available, human experts remain indispensable as their domain knowledge is crucial to identify and analyze intricate attack patterns [2]. Therefore, we need a way to make the heterogeneous data at hand available for domain experts. Visualizations offer a well-known path to achieve this goal. A visual representation can help a domain expert make sense of the displayed information and efficiently draw conclusions [12]. These observations lead to our work's research question:

RQ. *What are appropriate visualizations to assist security experts in detecting DLT threats?*

In this work, we make a two-fold contribution to this research question. We first characterize the domain problem: monitoring permissioned DLTs for attacks. This domain problem and derived general design requirements serve as the foundation for our visualization approach. The second part of our contribution is the task-centered design and prototypical implementation of *HyperSec*, a visual representation of security-relevant DLT information to support security experts' monitoring tasks.

The remainder of this work is structured as follows. Section 2 gives a brief overview of related academic work in the field of security visualizations in the blockchain domain. In Sect. 3, we flesh out the domain problem faced by security experts monitoring permissioned blockchain environments for immediate threats. Section 4 then introduces our visualization design and its prototypical implementation using open source technologies. Afterwards, we evaluate our visualization design by simulating attacks in Sect. 5. We discuss how an expert may proceed after an attack has been detected in Sect. 6. Finally, Sect. 7 concludes our work with a summary and possible future research directions.

2 Related Work

Recently, Tovanich et al. [23] carried out a systematic review to structure existing work on the visualization of blockchain data. Their research and previously conducted studies [19] identify several visualization approaches with a focus on criminal and malicious activity [8,13]. These surveys highlight that visualization tools for blockchains are on the rise. However, most of these existing visualization approaches for criminal or malicious activities in blockchains focus on historical analysis, i.e. detecting the events only after they have occured [23].

To effectively prevent attacks upfront, blockchain networks have to be actively monitored by blockchain security experts. Several studies discuss external and internal threats that could impair a blockchain network's functionality [9,18].

Zheng et al. propose a framework for monitoring the Ethereum blockchain's performance [24]. They introduce some respective metrics while using both node logs and Remote Procedure Calls (RPC) to gather data. Threat indicators to detect malicious activities in a blockchain network have recently been introduced by Putz et al. [17]. Based on this limited body of work from academia, blockchain metrics and threat indicators need to be made available to security experts for effective monitoring. Existing monitoring solutions like the dashboard by Bogner [3] focus only on transaction activity but do not consider other security-relevant data and metrics.

An approach pointing in this direction is the Hyperledger Explorer [21], the Hyperledger project's tool for monitoring Hyperledger blockchains. The Explorer connects to a local blockchain node and extracts data about blocks, transactions, peers, and more into a local PostgreSQL database. Additionally, a web application is available for inspecting blockchain data, including some basic visualizations of transaction data. However, these visualizations are not tailored to provide the necessary insights or indicators to detect threats. In addition, there is a Hyperledger Labs project integrating Fabric with ElasticSearch and Kibana, resulting in a Kibana dashboard able to display some transaction data [1]. Unfortunately, their visualizations are not very well suited to detecting blockchain threats in Hyperledger Fabric. In our experiments we found that the necessary integration and aggregation of additional data sources and custom visualizations are difficult to achieve in standard products like the Elastic stack.

Analyzing related work highlights an evident lack of dedicated and security-specific visualization approaches enabling security experts to monitor blockchain networks in real-time, while detecting common indicators of compromise or ongoing attacks on the network. Our work contributes a valuable solution approach to this issue.

3 Blockchain Security Monitoring

This Section addresses the first part of our contribution. Within our main contribution, we follow the user-centered and problem-driven Nested Blocks and Guidelines model (NBGM) for visualization designs [14,16]. This allows us to identify and address security experts' core problems and lay a foundation for a visualization design fitting their needs.

The first step of the NBGM is the definition of a domain problem. We characterize the problem at hand based on two primary sources of information. First, we consider reports from blockchain security professionals [11]. Second, we analyze literature on blockchain attacks to identify concerns for operators of a blockchain node [7,9,18]. We begin by outlining the overall blockchain security monitoring process in Sect. 3.1. The domain problem is then specified according to Miksch and Aigner's design triangle through more in-depth descriptions of specific users (Sect. 3.2), their tasks (Sect. 3.3), and data elements (Sect. 3.4) [15]. We address the second step of the NBGM (*Data/Operation Abstraction*) in

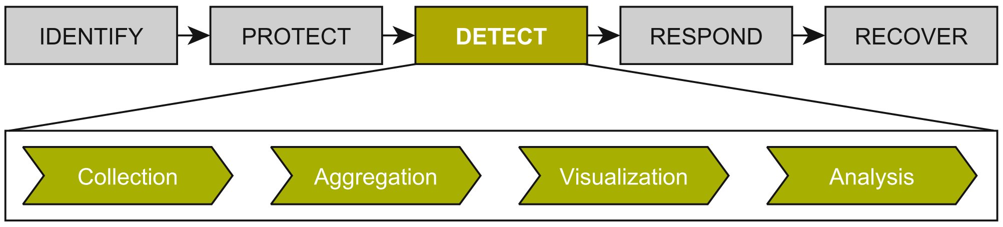


Fig. 1. Blockchain Security Monitoring process based on the NIST Cybersecurity Framework [5].

Sect. 3.5 by deriving general design requirements for a visualization approach to support blockchain security monitoring.

3.1 Blockchain Security Monitoring Process

Before we dive into users, tasks, and available data, we first need to understand the overall process underlying blockchain security monitoring. This subsection introduces our conceptual process based on the NIST Cybersecurity Framework for protecting critical infrastructures [5]. As shown in Fig. 1, the framework has five main functions: *Identify*, *Protect*, *Detect*, *Respond* and *Recover*. We apply these functions to a permissioned blockchain network. The *Identify* function serves to identify relevant assets and risks. This problem has been already addressed in prior work [17]. *Protect* involves a variety of protection measures applied to the system: identity management and access control, data security, secure configuration, and backups/log files, among others. These protection measures are usually part of the blockchain framework itself, with additional measures being applied at deployment time (such as secure configuration and appropriate backup procedures) [22]. The *Detect* function currently lacks appropriate visualization and analysis tools. It's the focus of this work and further developed in the following subsection. During the *Respond* phase, threats detected using our visualization approach are met with a response plan and appropriate mitigation actions. Finally, the *Recover* function provides appropriate tools to restore functionality after an attack has occurred. *Respond* and *Recover* are not specifically part of this work as attacks need to be identified before effective *Respond* and *Recover* can take place. Corresponding tools might be integrated into future work to permit swift threat response.

The *Detect* function can be subdivided into four smaller process steps. Relevant data needs to be collected (*Collection*) and aggregated to provide appropriate metrics if necessary (*Aggregation*). Data and metrics can then be visualized (*Visualization*) allowing domain experts to identify possible threats (*Analysis*). Please note that all steps beside *Analysis* can be performed automatically.

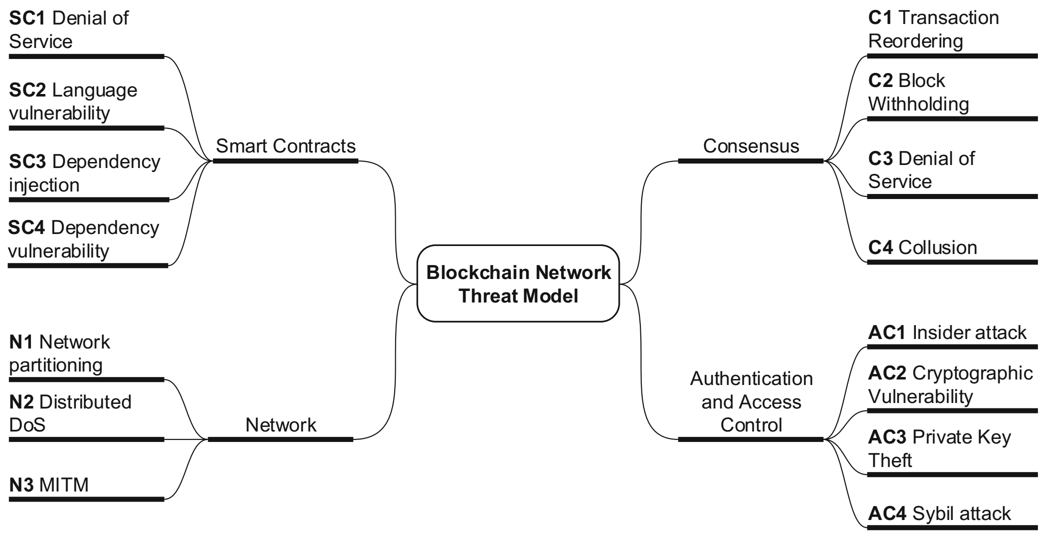


Fig. 2. Blockchain Networks Threat Model in attack tree notation.

3.2 Users

The intended users of visualization designs within the *Detect* function in the blockchain monitoring process are domain experts. These experts are responsible for analyzing blockchain data to identify malicious events within this function [11]. More specifically, we define the domain experts as security professionals knowledgeable in the cybersecurity domain. Therefore, we expect them to have the expertise to decide whether specific events or event series indicate an imminent threat to the blockchain. Within a permissioned blockchain, these security experts are responsible for monitoring the distributed network through the view of the local organization's blockchain node. Other organization's nodes could also be monitored, but data availability is likely limited due to access restrictions within the blockchain network.

3.3 Tasks

Visualizations or any other tool supporting the *Detect* function of the blockchain security monitoring process should be based on the tasks that the respective users need to carry out. Following the user characterization above, we derive the crucial tasks of the domain expert's work.

To illustrate the monitoring task's complexity, we show an overview of possible attacks in an attack tree notation in Fig. 2. The listed attacks are based on prior work [17,18] and related literature surveys [7,9]. For each leaf on the tree, there are various ways to successfully deploy the attack, which we did not include for conciseness. The attack tree focuses on the blockchain network and nodes. Therefore, it does not include application-specific attacks such as web

Table 1. Security expert tasks and related attacks (cf. Fig. 2).

Task	Description	Related attacks
T1	Identify vulnerable smart contracts	SC1, SC2, SC3
T2	Identify blockchain framework vulnerabilities	SC4, AC2
T3	Inspect log files of running services on demand	SC4, N1, N3, C3, C4
T4	Review networking activity	N1, N2, N3
T5	Compare transaction metrics over time	N2, C2
T6	Explore block and transaction history	SC1, SC2, C3, AC1
T7	Review configuration changes	C1, AC1
T8	Detect identity abuse	AC1, AC3, AC4

application vulnerabilities. Each of the shown attacks is indicated by different combinations of threat indicators [17]. Security experts need to identify threats based on these indicators as part of the *Analysis* process step. Visualizing the indicators provides the necessary overview to identify vulnerable components for in-depth analysis. Therefore, domain experts' overarching task is the *analysis of blockchain data to identify possible threats*, which is to be supported by visualizations. To allow domain experts to execute this work adequately, we have identified more specific tasks based on the attacks and corresponding threat indicators from prior work [17]. These tasks are shown in Table 1.

Each task comprises several sub-tasks that help accomplish the main task. To identify vulnerable smart contracts (*T1*), the expert may manually inspect smart contract code or scan smart contracts for vulnerabilities and inspect scan results. Identifying framework vulnerabilities (*T2*) can be accomplished by reading release notes for the framework and its dependencies. Since many anomalies can have multiple causes (i.e., low transaction throughput), log file inspection (*T3*) helps to identify the root cause of anomalies. To review networking activity (*T4*), the main indicators are the count of active connections to other peers, the activity level of those connections, and last seen times of offline peers. Transaction metrics (*T5*) include throughput, latency and unprocessed transactions. Block and transaction history monitoring (*T6*) implies watching the chain of blocks for inconsistencies such as changed blocks or missing transactions. Reviewing configuration changes (*T7*) includes both active and proposed changes to be able to intervene in case of manipulation attempts. Identity abuse (*T8*) is possible during all phases of an identity's lifecycle, so an expert must monitor issuance, usage in transactions, and revocation.

3.4 Data Elements

Blockchain Frameworks such as Ethereum and Hyperledger Fabric offer a number of data sources for monitoring. The most obvious data sources are blocks and associated transaction data [23]. These can be used to derive active users, smart contracts, and the general level of activity on the network (i.e., transaction throughput). Numerical data on network activity is also provided through metrics, which can be used to raise alerts for anomalous behavior. On a more technical level, each component of the blockchain node also provides log files. These files give detailed information about smart contract execution, consensus protocol violations, and other node internals. They can be helpful to determine the root cause of an anomaly.

3.5 Design Requirements

To wrap up this first part of our contribution, we derive the following general requirements for visualizations aiming to support the *Detect* function of the blockchain security monitoring process. The requirements are based on the above user, task, and data characterizations. Although we follow these requirements in the remainder of this work to design our prototype, they can serve as a general collection for respective visualization designs. We summarize the requirements under several main views that a Visual Analytics system supporting the domain experts' tasks should comprise:

R1 - General Security Information: A view should allow users to overview a series of general, security-relevant information from the monitored blockchain. Attention should be drawn to any changes on the blockchain's overall configuration (*T7*). Whenever new smart contracts are deployed to the blockchain, they should be checked (automatically or manually) for vulnerabilities. The results of these checks need to be made available for the analysts (*T1*). Additionally, newly discovered vulnerabilities within the applied blockchain framework should be shown to users within this general view (*T2*).

R2 - Network View: Another view should provide access to any data and metrics related to the peers and their network activities. This includes displaying available information about the peers themselves and the respective identities that interact with the blockchain on behalf of the peers (*T8*). This view should also provide visual access to any network-related metrics that assess the overall network's health (*T4*).

R3 - Transaction View: Domain experts need to access a view displaying information about the blocks and transactions being handled by the blockchain. This includes detailed information on the blocks and transactions themselves (*T6*) as well as a time-based view on transaction-related metrics allowing to identify any changes in typical structure and processing of transactions (*T5*).

R4 - Interactivity and Details: Any of the previously mentioned views (R1–R3) needs to be fully interactive to provide the best possible support for domain experts' tasks and enable exploratory analysis. Whenever suspicious actions or threat indicators are identified, experts also need access to further details and underlying log files (*T3*).

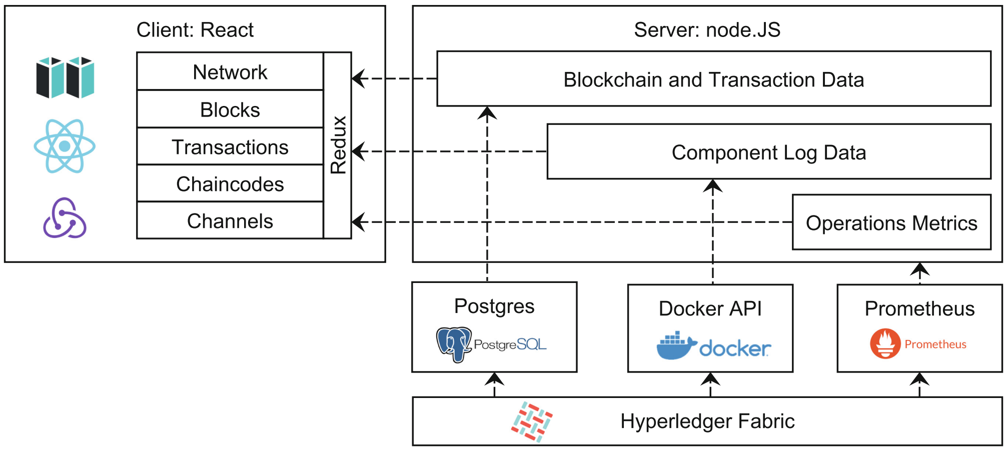


Fig. 3. Prototype architecture and data flows.

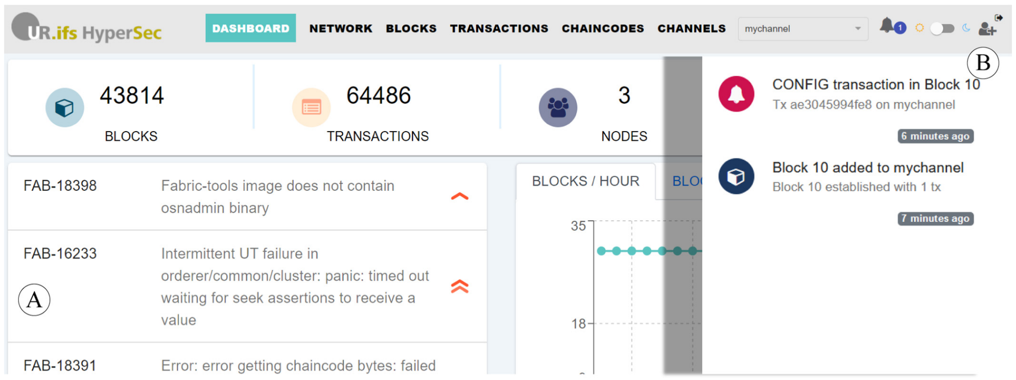
4 HyperSec: Hyperledger Security Monitoring Using Visual Analytics

We now introduce our prototype **HyperSec** (**Hyper**ledger **Sec**urity Explorer), a modified version of the open-source project Hyperledger Explorer based on the design requirements introduced in Sect. 3.5. The prototype is open-source and available online, along with a demo deployment[1]. Our modifications address the two remaining layers of the NBGM by designing our solution based on the domain problem and implementing it within a prototype.

4.1 Architecture and Technology

We choose Hyperledger Explorer as a starting point since it already provides a working synchronization architecture based on Hyperledger Fabric's block event subscription. We extend the existing architecture to allow for more comprehensive accessibility of relevant data and effective security monitoring. This results in the architecture displayed in Fig. 3. We keep the basic structure (data sources, server, and client) of the original architecture for interoperability and transparency reasons. However, in our previous study [17] we found that security-relevant information for Hyperledger Fabric must be retrieved from several data sources: the Hyperledger Fabric SDK, operations metrics, and the application logs available via Docker. Block data is already stored in Hyperledger Explorer's PostgreSQL database. We integrate additional metrics and log sources through server-side proxies to the respective Prometheus and Docker APIs. The React client accesses these through the API exposed by the Hyperledger Explorer server.

[1] https://github.com/sigma67/hypersec.



FAB-18398 Fabric-tools image does not contain osnadmin binary

FAB-16233 Intermittent UT failure in orderer/common/cluster: panic: timed out waiting for seek assertions to receive a value

FAB-18391 Error: error getting chaincode bytes: failed

CONFIG transaction in Block 10
Tx ae3045994fe8 on mychannel

Block 10 added to mychannel
Block 10 established with 1 tx

Fig. 4. *Dashboard* view: Security issues, alerts and general overview.

We implement the views defined in Sect. 3.5 by adapting existing views from the Hyperledger Explorer project. This allows us to retain the frontend structure while introducing new monitoring capabilities. Therefore, domain experts do not need to work with a completely new interface but rather get additional relevant information on the respective views. The updated views host a series of interactive visualizations based on the *visx*[2] visualization primitives for React. They all follow a similar structure: relevant data is retrieved from the client's *Redux* state handling, transformed for use in the visual display, mapped into visual primitives, and finally rendered [6].

4.2 Visual Representations and Interactions

We now go into more detail on our HyperSec prototype's visual representations addressing the requirements *R1*–*R3* and their interactivity (*R4*). As mentioned before, we integrate the visualizations into existing Hyperledger Explorer views to retain the familiar structure for domain experts. This Section is structured accordingly to the naming of the original Hyperledger Explorer views.

Views *Dashboard* and *Chaincodes*: To fulfill the Design Requirement *R1*, we adjust two views of the Hyperledger Explorer. First off, directly on the Explorer's landing page, called "Dashboard", we show a list of known Hyperledger Fabric issues of High/Highest importance from the Hyperledger JIRA[3] ordered by last updated (Fig. 4**A**). Any list item can be expanded to reveal additional information about the issue. Although there is no issue category directly reflecting security issues, this information is highly relevant for *T2 – Identify blockchain framework vulnerabilities*. Additionally, there is no other source for the respective information. In the side menu (Fig. 4**B**), an alert appears whenever the configuration of the monitored Hyperledger Fabric blockchain is changed (*T7*).

To allow domain experts to detect vulnerable chaincodes, we include available security scans in the respective "Chaincodes" view. Whenever a smart contract

[2] https://airbnb.io/visx/.
[3] https://jira.hyperledger.org.

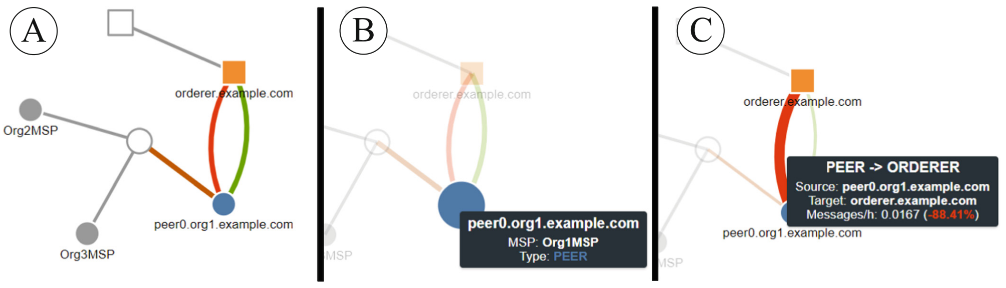


Fig. 5. *Network* view: Interactive visualization of network traffic between peers and orderers.

went through a security scan, analysts can directly check this scan's results in the HyperSec prototype (*T1*). We use the open-source static analysis tool revive-cc[4] to detect security vulnerabilities and store the scan result in the Hyperledger Explorer PostgreSQL database. To ensure the scans are up to date, we set up automated jobs to regularly generate security reports of deployed chaincodes.

View *Network*: The *Network* view targets design requirement *R2* intended for tasks *T4* and *T8*. The original Hyperledger Explorer shows a tabular list with basic information about the peers connected to the monitored Hyperledger Fabric network. In our HyperSec prototype, we extend this table with a force-directed node-link diagram to effectively visualize networking activity (Fig. 5**A**). The nodes' different shapes indicate different peer types within the network: Circles are used to display peers while rectangles represent orderers. Links between the glyphs are used to display known networking activities.

However, the unavailability of core information restricts this view's expressivity. While rich information about the peers can be easily retrieved from the Hyperledger Explorer, no data about the peers' network connections is provided. Therefore, HyperSec retrieves networking information directly from Hyperledger Fabric through the Prometheus API. By doing so, experts get at least some information about the peers' networking activity within the own Membership Service Provider (MSP). However, as the Hyperledger Fabric network is decentralized, it is not possible to get any information about other MSPs' networking activities. Because of this restriction, we introduce two empty nodes in the node-link diagram (uncolored nodes in Fig. 5**A**), which mark the border of the monitoring visibility regarding networking activities. Nodes within the owned MSP are colored; those within other MSPs are greyed out.

Links connecting the nodes in the graph represent known network connections. Again, outside the own MSP's borders, experts do not get much information. Therefore, we connect any foreign peer and orderer to the respective artificial node. The coloring of the links follows a continuous scale from -1 to 1. This scale measures the current deviation of the link's message traffic from the average, by comparing traffic in the last hour with traffic in the previous seven

[4] https://github.com/sivachokkapu/revive-cc.

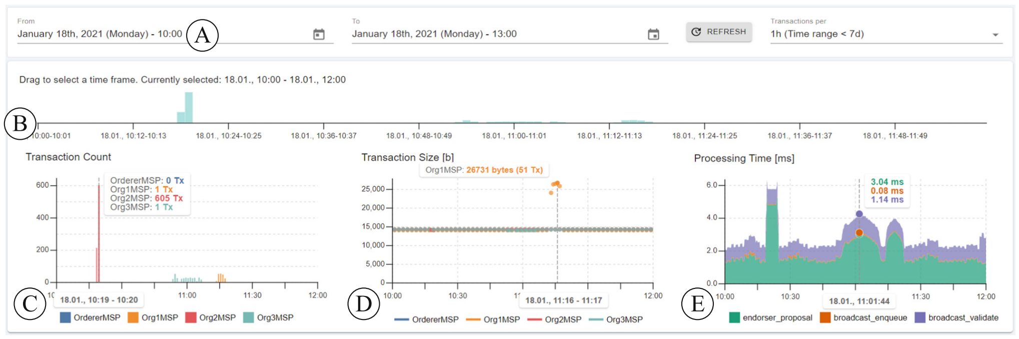


Fig. 6. *Transactions* view: Interactive visualizations for transaction count, size and processing time.

days. If this deviation is low, the link is colored in a green tone. A red link, in contrast, marks a high variation of message numbers.

The node-link diagram is fully interactive. Nodes are draggable to ensure that security analysts can adjust the layout to their own needs if necessary. Hovering over nodes (Fig. 5**B**) or links (Fig. 5**C**) highlights the hovered object and shows additional status information about it.

View *Transactions*: This view (*R3*) satisfies tasks *T5* and *T6*. Some modifications to the original simple table view ensure that the transactor identity and transaction size are visible. The primary adjustment we made to this view is introducing four visualizations (Fig. 6). To ensure a high performance of the visualizations even when dealing with several thousands of transactions, we implement an efficient data bucketing algorithm which allows easy and fast look up of relevant transaction data (see Algorithm 1).

We make small adjustments to the original timeframe selection (Fig. 6**A**). The selection defines the time range for which information about transactions should be displayed. On the right side of the timeframe selection, we added a dropdown menu to select the aggregation granularity (1 min, 1 h, 12 h, 24 h) for the visualizations. This helps security experts if they need to compare and contextualize available information.

The wide bar chart (Fig. 6**B**) always displays the entire selected date range. Each bar represents the number of transactions within a specific range of time specified through the aggregation granularity. This bar chart supports analysts in navigating the selected time range. A brushing interaction (horizontal dragging on the chart) selects an even smaller time range for detailed analysis. On interaction, the other visualizations (Fig. 6**C**, **D**, and **E**) and the transactions table are dynamically updated with data from this narrowed time range.

A stacked bar chart (Fig. 6**C**) visualizes the number of transactions per aggregation window. However, it does this only for the transactions selected through the brushing interaction on the visualization Fig. 6**B**. It shows the transaction count's composition based on which MSP contributed how many transactions. The scatterplot Fig. 6**D** shows the transaction size in bytes throughout the time

Algorithm 1: Transaction data bucketing

```
Input: Time Window from timestamp $t_s$ to $t_e$ with $t_s, t_e \in T$, $t_s < t_e$, and
       $T \in \mathbb{R}$. Aggregation granularity $s_b \in \mathbb{R}$
Output: Map $M^{tx}$ with transactions sorted into the respective time-based
        bucket
function generateTxBuckets($t_s, t_e, s_b$):
    $L^{tx} \leftarrow getTransactionListForTimeWindow(t_s, t_e)$;
    $M^{tx} \leftarrow$ new $Map()$;
    $t_b \leftarrow t_s$;
    while $t_b < t_e$ do
        $i_b \leftarrow \lfloor t_b / s_b \rfloor$;
        $b \leftarrow$ new $Bucket()$ ;    // Object for transactions and meta-data.
        $M^{tx}_{i_b} \leftarrow b$;
        $t_b \leftarrow t_b + s_b$;
    end while
    foreach $tx \in L^{tx}$;          // Find correct $Bucket$ and update with $tx$.
     do
        $i_{tx} \leftarrow \lfloor (t_{tx} - t_s)/s_b \rfloor$;
        $M^{tx}_{i_{tx}} \leftarrow M^{tx}_{i_{tx}} \cup parse(tx)$
    end foreach
    return $M^{tx}$;
end
```

range for each MSP. Each circle on the scatterplot represents the average size of transactions submitted by a specific MSP. This aggregation is performed to scale the chart for large numbers of transactions. During attacks, thousands of transactions can be submitted within just minutes, thus freezing the chart if each transaction were drawn individually. The stacked area chart Fig. 6**E** finally shows the development of three different metrics, which we identify as helpful to get an idea for the processing time in seconds that a transaction needs from proposal to validation. As information for processing times are not available distinctively per transaction but continuously per time unit, we choose to display this metrics with a continuous visualization technique.

The visualizations Fig. 6**C**, **D**, and **E** are again fully interactive. Hovering individual bars or hovering along the continuous sizes and times displays additional information as tooltips. Different metrics can also be toggled using the legend icons below the visual representations.

5 Evaluation

For our evaluation, we focus on three common attacks that cover the majority of the tasks outlined in Table 1: **SC2**, **N2**, and **AC1**. We simulate these attacks a Hyperledger Fabric test network, which the HyperSec prototype is connected to.

SC2 refers to a language vulnerability, i.e. a software bug that exposes chaincode to malicious exploits. A security expert may become aware of such an exploit by identifying vulnerable smart contracts (*T1*) and by inspecting transaction history (*T6*). For example, consider a read-after-write vulnerability detected by the chaincode scanner *revive-cc*. The security expert can inspect an automatically generated chaincode scan in the *Chaincodes* view. Intuitively, the experts check for past exploitations using the *Transactions* view. Thereto, the transactions table can be filtered using the chaincode name and applicable time frame. The filtered transactions can be inspected individually to find unusual read/write sets.

N2 refers to a distributed denial of service attack. If a peer or orderer is targeted by a traffic-based denial of service attack, its connection to other peers will be impaired as well. The *Network* view (Fig. 5**C**) shows high deviation in gossip communication traffic to the targeted peer during such an attack (*T4*). If the local peer is targeted, the metrics in the *Transaction* view (Fig. 6**E**) show increased transaction processing latency due to high peer load (*T5*). For attackers that can send transaction to the network, transaction-based DoS is more effective. Figure 6**C** and 6**D** show two such attempts using high transaction volume (**C**) and large transaction size (**D**). Figure 6**E** also shows spikes in processing latency during the time of attack (spikes 1 and 3 in that chart).

To investigate the source of the anomaly, experts can check the peer logs, which are available in the Network view (*T3*). They cross-reference any error messages with open issues in the Hyperledger JIRA, which are available in the Dashboard view (*T2*).

AC1 refers to an attack where an insider abuses valid credentials for malicious purposes. Consider an insider attempting to corrupt the blockchain network's configuration using a configuration transaction. Security experts are immediately notified about the configuration change in the notification sidebar (*T7*, see Fig. 4). Details of the attempted configuration change are available in the transaction history table (*T6*), where the full read-write set of the transaction is available by selecting the respective transaction.

6 Discussion

The evaluation has shown that the visualizations can assist a security expert in detecting ongoing attacks. If an attack is detected, the next steps in the Cybersecurity Framework (see Fig. 1) are *analysis*, *respond* and *recover* activities, which are discussed hereafter.

Analysis. Based on the present threat indicators the expert then proceeds with *analysis* of the root cause. The logs shown in HyperSec can be a starting point, but may only show symptomatic errors. In-depth analysis of application and network logs on the systems running blockchain components can yield further information. The expert must determine if it is a crash fault or a byzantine fault. At the same time, a communication channel should be available with other organizations of the consortium to determine if it is a more widespread problem. Guidelines and checklists can help structure this process.

Respond. Once the cause is identified, the expert contacts operations teams to request mitigation actions. Local or network configuration changes can mitigate crash faults and network/consensus threats (see Fig. 2). Compromised smart contracts may require an upgrade, or even a ledger rollback if the consequences were severe. Hyperledger Fabric supports ledger snapshots for this purpose [22]).

Recover. After mitigation of an attack, evidence collection is another subject of interest. System and Docker logs are the primary source of evidence, complemented by ledger transaction data stored in HyperSec's PostgreSQL database. However, the forensic analysis of attacks on Hyperledger Fabric is a topic in need of further research.

7 Conclusion

This work introduced the task-oriented design and prototypical implementation of HyperSec, a visual analytics security monitoring tool tailored for Hyperledger Fabric. Throughout the design of HyperSec, we followed the NBGM design methodology. The domain problem describes the activities of the blockchain security monitoring process to be supported by visualizations. Subsequently, we identified the involved users, their specific tasks, and the available data elements. These considerations culminated in design requirements that apply to any visualization system aiming to support blockchain security analysts. Our prototype HyperSec picks up on these design requirements. It extends the open-source architecture of Hyperledger Explorer with additional security-relevant data sources. The data is aggregated, processed and displayed in appropriate visualizations supporting blockchain security analysts to detect potential attacks.

Our prototype might not cover every possible subtask of the defined tasks of blockchain security analysts. This is in part due to limited availability of data provided by Hyperledger Fabric itself. We plan to update our prototype as additional data sources become available in the future, and are open to contributions from the community.

The security of the monitoring tool itself is also important, as it should not contribute additional attack vectors by leaking blockchain data. During our implementation we found some bugs and vulnerabilities within Hyperledger Explorer, which we subsequently fixed and contributed to the upstream project.

References

1. Baset, S., Prehoda, B.: Hyperledger Labs Blockchain Analyzer, March 2021. https://github.com/hyperledger-labs-archives/blockchain-analyzer. 30 May 2019
2. Ben-Asher, N., Gonzalez, C.: Effects of cyber security knowledge on attack detection. Comput. Hum. Behav. **48**, 51–61 (2015). https://doi.org/10.1016/j.chb.2015.01.039

3. Bogner, A.: Seeing is understanding: anomaly detection in blockchains with visualized features. In: Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing, New York, NY, USA, pp. 5–8. ACM (2017). https://doi.org/10.1145/3123024.3123157
4. Boshmaf, Y., Al Jawaheri, H., Al Sabah, M.: BlockTag: design and applications of a tagging system for blockchain analysis. In: Dhillon, G., Karlsson, F., Hedström, K., Zúquete, A. (eds.) SEC 2019. IAICT, vol. 562, pp. 299–313. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22312-0_21
5. Calder, A.: NIST Cybersecurity Framework (2018). https://doi.org/10.2307/j.ctv4cbhfx
6. Chi, E.: A taxonomy of visualization techniques using the data state reference model. In: Proceedings of the IEEE Symposium on Information Visualization 2000, pp. 69–75. IEEE Computer Society (2000). https://doi.org/10.1109/INFVIS.2000.885092
7. Dabholkar, A., Saraswat, V.: Ripping the fabric: attacks and mitigations on hyperledger fabric. In: Shankar Sriram, V.S., Subramaniyaswamy, V., Sasikaladevi, N., Zhang, L., Batten, L., Li, G. (eds.) ATIS 2019. CCIS, vol. 1116, pp. 300–311. Springer, Singapore (2019). https://doi.org/10.1007/978-981-15-0871-4_24
8. Di Battista, G., Di Donato, V., Patrignani, M., Pizzonia, M., Roselli, V., Tamassia, R.: Bitconeview: visualization of flows in the bitcoin transaction graph. In: 2015 IEEE Symposium on Visualization for Cyber Security (VizSec), pp. 1–8. IEEE (2015). https://doi.org/10.1109/VIZSEC.2015.7312773
9. Homoliak, I., Venugopalan, S., Reijsbergen, D., Hum, Q., Schumi, R., Szalachowski, P.: The security reference architecture for blockchains: towards a standardized model for studying vulnerabilities, threats, and defenses. IEEE Commun. Surv. Tutor. (2020). https://doi.org/10.1109/COMST.2020.3033665
10. Jensen, T., Hedman, J., Henningsson, S.: How TradeLens delivers business value with blockchain technology. MIS Quart. Execut. (2019). https://doi.org/10.17705/2msqe.00018
11. Kacherginsky, P.: Attacking and Defending Blockchain Nodes. In: DEFCON 2020, p. 54 (2020)
12. Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., Melançon, G.: Visual analytics: definition, process, and challenges. In: Kerren, A., Stasko, J.T., Fekete, J.-D., North, C. (eds.) Information Visualization. LNCS, vol. 4950, pp. 154–175. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70956-5_7. iSSN: 03029743
13. McGinn, D., Birch, D., Akroyd, D., Molina-Solana, M., Guo, Y., Knottenbelt, W.J.: Visualizing dynamic bitcoin transaction patterns. Big Data **4**(2), 109–119 (2016). https://doi.org/10.1089/big.2015.0056
14. Meyer, M., Sedlmair, M., Quinan, P.S., Munzner, T.: The nested blocks and guidelines model. Inf. Vis. **14**(3), 234–249 (2015). https://doi.org/10.1177/1473871613510429
15. Miksch, S., Aigner, W.: A matter of time: applying a data-users-tasks design triangle to visual analytics of time-oriented data. Comput. Graph. **38**, 286–290 (2014). https://doi.org/10.1016/j.cag.2013.11.002
16. Munzner, T.: A nested model for visualization design and validation. IEEE Trans. Visual Comput. Graphics **15**(6), 921–928 (2009). https://doi.org/10.1109/TVCG.2009.111
17. Putz, B., Pernul, G.: Detecting blockchain security threats. In: 2020 IEEE International Conference on Blockchain (Blockchain), pp. 313–320. IEEE (2020). https://doi.org/10.1109/Blockchain50366.2020.00046

18. Putz, B., Pernul, G.: Trust factors and insider threats in permissioned distributed ledgers. In: Hameurlain, A., Wagner, R. (eds.) Transactions on Large-Scale Data- and Knowledge-Centered Systems XLII. LNCS, vol. 11860, pp. 25–50. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-60531-8_2
19. Sundara, T., Gaputra, I., Aulia, S.: Study on blockchain visualization. Int. J. Inform. Visual. **1**(3), 76–82 (2017). https://doi.org/10.30630/joiv.1.3.23
20. The Linux Foundation: DLTLabs Case Study - Hyperledger (2020). https://www.hyperledger.org/learn/publications/dltlabs-case-study
21. The Linux Foundation: Hyperledger Explorer (2020). https://www.hyperledger.org/use/explorer
22. The Linux Foundation: Hyperledger Fabric 2.3 Documentation (2020). https://hyperledger-fabric.readthedocs.io/en/release-2.3
23. Tovanich, N., Heulot, N., Fekete, J., Isenberg, P.: Visualization of blockchain data: a systematic review. IEEE Trans. Visual. Compute. Graphics 1 (2019). https://doi.org/10.1109/TVCG.2019.2963018
24. Zheng, P., Zheng, Z., Luo, X., Chen, X., Liu, X.: A detailed and real-time performance monitoring framework for blockchain systems. In: Proceedings - International Conference on Software Engineering (2018). https://doi.org/10.1145/3183519.3183546, iSSN: 02705257

3 RQ3: Secure Information Sharing in a DLT Consortium

3.1 Comparing Successful DLT Consortia: A Lifecycle Perspective. [P8]

Status:	published
Publication:	55th Hawaii International Conference on System Sciences 2022
Submitted:	15 June 2021
Revised:	05 September 2021
Accepted:	12 September 2021
Citation:	Benedikt Putz and Günther Pernul. 2022. Comparing Successful DLT Consortia: A Lifecycle Perspective. In 55th Hawaii International Conference on System Sciences 2022, 4591–4600.

Conference Description: Since 1968, the Hawaii International Conference on System Sciences (HICSS) has been known worldwide as the longest-standing working scientific conferences in Information Technology Management. HICSS provides a highly interactive working environment for top scholars from academia and the industry from over 60 countries to exchange ideas in various areas of information, computer, and system sciences. HICSS is the #1 Information Systems conference in terms of citations as recorded by Google Scholar.

Proceedings of the 55th Hawaii International Conference on System Sciences | 2022

Comparing Successful DLT Consortia: A Lifecycle Perspective

Benedikt Putz
University of Regensburg
benedikt.putz@ur.de

Günther Pernul
University of Regensburg
guenther.pernul@ur.de

Abstract

In 2021, enterprise distributed ledger technology has evolved beyond the proof-of-concept stage. It is now providing business value to large consortia in several successful and well documented case studies. Nevertheless, other consortia and initiatives are stuck in early stages of consortium formation or conceptualization. They stand to benefit from lessons learned by successful consortia, but an in-depth comparison has not yet been conducted. Thus, this study performs the first methodological comparison of large DLT consortia that have launched a product. Based on the temporal evolution of these consortia, a lifecycle with 4 stages and 12 sub-phases is developed to provide further guidance for early-stage consortia. The results show how 9 pioneer consortia have successfully integrated novel DLT into existing processes, but also point out challenges faced on the way.

1. Introduction

Distributed Ledger Technology (DLT) offers great potential to improve inter-organizational processes by improving information transparency, traceability, efficiency and ultimately reducing costs [1, 2]. Despite the promised benefits, it has not yet reached widespread adoption. DLT faces numerous technical challenges with regard to scalability, privacy and interoperability among others [3]. But these are not the main issue preventing adoption. Indeed, the main challenge for businesses looking to reap DLT's benefits is successful collaboration with ecosystem partners [4, 5]. A global survey with 1400 respondents from businesses around the world in March 2020 further confirms this assessment [6]. 30% of its respondents agree to face challenges in forming a consortium for DLT-based collaboration. In addition, 40% face at least one of several challenges participating in the consortium collaboration (such as defining balanced governance rules, defining roles and responsibilities, cross purposes of members). Consequentially, despite significant progress in some industries, many initiatives remain stuck in a proof of concept stage and are thus investing without gaining business benefits. Still, there are several pioneers that have used the technology successfully to improve business processes. To date, researchers have only published isolated studies of such cases [7, 8, 9]. Going beyond single case studies, multiple case study designs can provide more robust results by revealing findings through replication logic [10]. These findings may prove useful to newer DLT consortia in early stages of development. To our knowledge, no such study has been conducted with regard to DLT consortia. Therefore, we derive the following research questions:

RQ1: *What are the commonalities and differences between operational DLT consortia?*

RQ2: *Which phases does a DLT consortium undergo during its lifecycle?*

Contribution. To answer these questions, we conduct a systematic multiple case study of successful production use cases of DLT in a consortium setting. To our knowledge, this is the first systematic study of DLT consortia that have launched an operational network. Besides determining commonalities and differences, we also focus on the temporal dimension of consortium building by developing a lifecycle from empirical evidence. We contribute to theory by aggregating and comparing evidence from distinct DLT case studies, while building new theory for future empirical studies with the lifecycle. Our study contributes to practice by providing a guideline for DLT consortia in early adoption stages, which allows them to tackle governance issues in a structured way.

This paper is organized as follows: Section 2 provides an overview of related research with regard to consortium collaborations and blockchain governance. Section 3 explains our methodology for case study selection and lifecycle development. It also provides reasoning for the selection criteria mentioned in the previous paragraph. In Section 4, the selected cases are briefly introduced. Based on an in-depth review of each case, Section 5 focuses on answering RQ1, while Section 6 introduces the DLT consortium lifecycle answering RQ2. Finally, we discuss our findings and limitations in Section 7 and wrap up with a summary of the results in Section 8.

2. Background

While literature commonly uses the term blockchain, this paper uses the term DLT, which covers a broader range of technical frameworks (i.e. R3 Corda is technically not a blockchain [11]). We also use the term *consortium* for a DLT-based partner network, defined as *"a form of cooperation between institutions that see value in sharing resources and know-how to save costs"* [7]. In particular, business-oriented consortia focus on solving business problems with DLT, as opposed to technology-oriented consortia, which focus on developing DLT platforms (i.e. R3, Hyperledger). This study focuses on business-oriented consortia.

Before explaining our methodology, we give some background on information sharing in partner networks, which is currently the main business objective of DLT consortia. Regarding blockchain and DLT, the research stream blockchain governance focuses on the processes and decision rights in DLT consortia and is thus closely related to the present work.

2.1. Information sharing in partner networks

A network of collaborating business partners is the foundation necessary to begin information sharing with a technological artifact. Such partner networks are typically formed in stages, which are identified by Larson et al. and shown in Figure 1 [12]. The process starts with the **trial period**, where companies exchange initial information about each other's knowledge, capabilities and resources. First experiences are made based on initial collaborations, thus generating trust. Once enough trust has been built up, companies proceed to an **integration period**, where companies begin having increasing influence over others in their interactions. More risks are taken and short-term losses are more easily accepted with the expectation of long-term profits. In this period, processes are more closely aligned to reduce organizational differences [13]. This also includes closer integration of technical infrastructure, which improves the effectiveness of the collaboration by increasing transparency. Larson refers to this constant improvement as "Kaizen".

DLT is a technological artifact that can be used to significantly enhance the technical integration. DLT removes the need to trust one partner or outsider with data management, while providing non-repudiation,

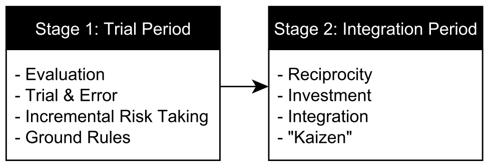


Figure 1: Stages of the business network formation process [12]

integrity and transparency guarantees [14].

Once partners are sufficiently integrated, the network is established (network phase). A functioning network is characterized by an atmosphere of trust, which enables open communication and collaborative problem solving. While DLT cannot guarantee this trust, it can enable it by providing a stable technological basis that even permits isolated misbehavior under an honest majority [15].

2.2. Blockchain Governance

First and foremost, blockchain governance research deals with the distribution of responsibilities and power among consortium participants. This includes decision-making rights and processes and accountability of decision-makers [16].

One stream of research focuses on developing layers and dimensions for governance of blockchains [17, 18]. Pelt et al. focus mostly on permissionless blockchains (public networks without access restrictions) and their open-source governance. They describe three layers: Off-chain community, off-chain development and on-chain protocol. Dimensions further describe the aspects of each layer [17]. The approach by Beck et al. instead focuses on permissioned networks (closed networks focused on enterprise usage), establishing incentives, accountability and decision rights as core dimensions for blockchain governance [18].

Ziolkowski et al. examine decision problems in blockchain governance, providing guidance on the handling of challenging decisions faced by the interview participants [16, 2]. Their case study interviews were conducted in 2017, so most of the cases were still creating proof-of-concept prototypes. In addition, there are many case studies focusing on the development and challenges faced by a single consortium, i.e. TradeLens [8], Cardossier [7] and MediLedger [9]. The University of Cambridge conducted its latest benchmarking study on live DLT networks in 2019 [19]. However, the study is missing a transparent and methodological approach to consortium selection. The lack of consideration for consortium size leads to skewed results in two ways. Firstly, the study itself categorizes 77% of its database as "blockchain memes", i.e. systems without real multi-party consensus. Secondly, results are biased toward individual companies developing solutions (71%) instead of consortia.

Few existing studies consider the temporal dimension of consortium development. Especially regarding permissioned blockchains, there is still uncertainty regarding the formation and development processes of blockchain consortia. The Cambridge Benchmarking Study proposes four stages (exploration, concept, trial, production), but does not detail how these stages were determined or relate them back to individual cases [19]. The study also does not investigate in detail what happens in each phase and what happens beyond the initial production deployment. Deventer et al. created a strategic options model with exploration and exploitation phases, but this was only based on a conceptual experiment [20]. Therefore, this study fills the gap by methodologically deriving a holistic lifecycle of blockchain consortium development from empirical evidence, from consortium formation to post-launch expansion.

3. Methodology

Based on the initial model derived from business network formation (see Figure 1), we further refine this model based on a comparative multiple case study research design [10]. We perform a holistic study of multiple cases, where each case represents a different context in which DLT is being adopted. The study aims to answer our research questions RQ1 and RQ2, motivated by the lack of cross-case comparisons for DLT consortia in the literature.

We first conduct a literature review of existing DLT use cases to determine applicable cases for our study. To find as many cases as possible we conducted a breadth-first search, which included:

- DLT and industry news websites (LedgerInsights[1], Global Trade Review[2], Forbes Blockchain 50 [21]
- DLT framework websites (i.e. Hyperledger, R3, Multichain use case sections)
- industry review studies (WTO study on DLT in Global Trade [22])

Overall, the breadth-first search yielded a total of 35 DLT consortia in various stages of development. To deepen our view of each case, we retrieve available information from the web. We search for mentions in scholarly literature using the databases Google Scholar, IEEE Xplore and Scopus and combining the consortium brand name (column 1 in Table 1) with the term "case study". Other valuable sources include the consortium website, consortium partners' websites, press interviews and news articles.

A critical part of this work is defining a successful case. We consider a case successful *iff*

1. it has been operating live for at least 6 months
2. at least 7 participants operate DLT nodes

The period of live operation is determined based on the point of live deployment in the unified software development process [23]. We use 6 months as the lower bound to be able to investigate post-launch phases. 7 nodes is the lower bound to tolerate at least 2 dishonest nodes in the $3f + 1$ byzantine fault-tolerant consensus model [15]. As trust issues among participants are a key point on the decision path to DLT adoption [14], tolerating some misbehavior is essential to build a successful platform. In combination with 6 months of live operation, the lower bound on network nodes also serves to ensure that some business value is being delivered, as otherwise organizations would begin leaving to save costs.

Additionally, to ensure sufficient empirical evidence for analysis we define a structured set of criteria based on the well-known Information Quality (IQ) Framework [24]. We focus on Intrinsic IQ and Contextual IQ, where we consider the following characteristics:

Intrinsic IQ: accuracy, objectivity, believability, and reputation of data

Contextual IQ: appropriateness of contextual parameters of data for the task at hand

Therefore, Intrinsic IQ is based on the quality of associated publications. We consider a case's intrinsic IQ as *high* if there is at least one publication with Q1/Q2 ranking in ISI WoK Journal Rankings[3], or at least B quality in CORE conference/journal rankings[4]. We consider it *medium*, if there are at least 5 different sources of acceptable quality (white papers, online case studies, news articles, website entries).

Contextual IQ is used to measure the completeness of the case study based on publicly available data, as well as its appropriateness based on the consortium's success. This metric is computed as the percentage of

[1]`www.ledgerinsights.com`
[2]`www.gtreview.com`
[3]`http://www.webofknowledge.com/JCR`
[4]`https://www.core.edu.au/conference-portal`

Table 1: Overview of selected case studies (in alphabetical order).

Name	Lead	Partner	Platform	Sector	Nodes	Legal Form	Country
B3i Re	B3i Services	R3	Corda	Insurance	21	Stock Corp.	Switzerland
Bakong	NBC	Soramitsu	Iroha	CBDC	16	led by NBC	Cambodia
Cardossier	cardossier	AdNovum	Corda	Vehicles	8	Association	Switzerland
Contour	Contour	Contour	Corda	Trade Finance	8	Limited	Singapore
covantis	Covantis	ConsenSys	Quorum	Agriculture	18	SA	Switzerland
DL Freight	Walmart Canada	DLT Labs	Fabric	Logistics	30	Incorporated	Canada
MediLedger	Chronicled	Chronicled	Ethereum	Pharma	10	Incorporated	US
TradeLens	Maersk	IBM	Fabric	Logistics	14	Subdivision	US
we.trade	we.trade	IBM	Fabric	Trade Finance	16	DAC	Ireland

pertinent questions that we were able to answer based on available sources. Specifically, we consider the following questions:

Q1: Is sufficient information consortium evolution available (trial/integration phases)?

Q2: Did the consortium platform launch at least 6 months ago?

Q3: Do at least 7 participating organizations control DLT nodes?

Q4: Could the name of the used DLT platform be extracted?

Cases were included in our study if they have an intrinsic IQ of at least *medium* and a contextual IQ of 100%. After applying this filter, 9 cases remain, which we focus on in the following sections. The full table of gathered data on consortia is available online, along with intrinsic and contextual IQ filter results[5].

For lifecycle development, we follow a bottom-up, empirical and iterative process. Figure 1 from Section 2.1 provides initial guidance on the trial and integration phases. We subsequently refine these phases based on common milestones of each consortium (see Table 2) and activities occurring between these milestones (see Table 3). Finally, we performed several semi-structured expert interviews with consortium representatives to validate our findings.

4. Selected Cases

An overview of selected case studies is given in Table 1. Hereafter, a brief description of each selected case study is given, followed by a citation of the primary source(s) for the case.

[5]`https://drive.google.com/drive/folders/1M3046mb6X1UsxnR9l_dj80jXzJXRiivb`

B3i Reinsurance. B3i refers to the Blockchain Insurance Industry Initiative, a consortium jointly currently owned by 21 insurance companies. The consortium develops solutions for the insurance market, with its current solution B3i Reinsurance (B3i Re) focused on reinsurance Catastrophe Excess of Loss (Cat XoL) treaties. Primary source for B3i Reinsurance is the case study by its implementation partner R3 [25], accompanied by a large number of detailed blog posts on their website.

Bakong. Bakong is an initiative of the National Bank of Cambodia (NBC), and one of the first Central Bank Digital Currencies (CBDC) to launch in production on Hyperledger Iroha in June 2020. Its stated objective is improving financial inclusion across Cambodia through simplified access to bank accounts and near real-time mobile payments. Primary source is the Bakong Guide Book published by the NBC [26].

Cardossier. The Cardossier consortium was built with the goal of establishing a single source of truth for car-related data across a car's lifecycle. It currently has 20 members, including government organizations and businesses from the automotive, finance and insurance sectors. It runs in production on the Corda DLT since June 2020. The primary source is the case study by Zavolokina et al. [7]. In addition, this consortium is noteworthy for its double-digit count of published research papers, thanks to its collaboration with the University of Zurich.

Contour. Contour is a Trade Finance platform focused on digitizing Letters of Credit. It went live on an R3 Corda network in October 2020. R3 also provides a case study focused on Contour, which is our primary source [27].

Covantis. Covantis aims to digitize processes in agricultural trading and shipping by creating a trusted single source of information. It relies on the ConsenSys-supported Quorum blockchain platform. ConsenSys has published a case study that serves as our primary source [28].

DL Freight. DL Freight was built by DLT Labs for Walmart Canada to address freight invoicing issues for logistics carriers. The platform is based on Hyperledger Fabric and runs in production since October 2019. Primary source is a case study conducted by the University of Arkansas [29].

MediLedger. MediLedger is a US-based consortium focusing on product traceability and preventing counterfeits in the pharmaceuticals supply chain. It was founded in 2017, and went into production on Parity Ethereum in October 2019. Primary source is the case study by the University of Bamberg [9].

TradeLens. TradeLens is a supply chain platform focused on digitizing bills of lading for container shipping. It was founded by Maersk, starting with research prototypes of different names as early as 2013. The decision to commercialize and launch it as the TradeLens platform was made in December 2018. Since then, it has grown to become the largest DLT platform by ecosystem size, currently claiming more than 200 members globally. Primary source for TradeLens is the case study conducted by Jensen et al. [8].

we.trade. we.trade is a platform focused on trade finance, specifically on Bank Payment Undertakings (buyers providing bank payment guarantees to sellers). It was founded at the beginning of 2017 as a joint venture between 12 European banks and subsequently went into production in October 2018. Primary source is the case study by its technology provider IBM [30], complemented by interviews on Global Trade Review for more recent developments since 2018.

For brevity, we refer to the cases described in this chapter simply as *cases* or *consortia* in the following sections.

5. Case Comparison

In this Section, we attempt to answer our first research question based on the findings from the case studies: ***RQ1**: What are the commonalities and differences between operational DLT consortia?*

Specifically, we analyze six dimensions derived from blockchain governance literature and our own research: *Platform choice* [19], *Network size* [15], *Incentives* [17, 18], *Legal Form* [18], *Disintermediation* [31, 4] and *Interoperability* [19, 4].

Platform choice. The choice of DLT platform differs among consortia. Hyperledger Fabric is the most popular framework (4 consortia), closely followed by Corda (3 consortia). Ethereum and Hyperledger Iroha support 1 consortium each. Regarding the full list of 33 consortia, the distribution changes. 16 consortia use Hyperledger Fabric and only 6 Corda. Quorum is used by 5, with others (i.e. Multichain, Ethereum, Iroha) totaling 6. Corda is almost exclusively used by Trade Finance and Insurance consortia, with Cardossier marking the exception. Overall, Hyperledger Fabric seems to be the most flexible platform fitting most use cases, but other platforms have valid use cases as well.

Network size. The network size refers to the number of distinct independent organizations operating the underlying DLT network (i.e. taking on a validator role). The size of the selected consortia varies between 8 (Contour) and 30 (DL Freight). Member counts published in official press releases must be carefully considered. For example, TradeLens claims more than 200 ecosystem members, but only a small number of those are actually performing validator roles on the DLT. For TradeLens, validators are referred to as “trust anchors”, a role currently assumed by ocean carriers. 14 ocean carriers and an additional node operated by consortium’s legal entity perform consensus validation for the network. Other ecosystem partners have much less control and only interact with the DLT via an API.

Incentives. Initially, incentives refer to reasons for new members to join the network. For marketing these incentives, the consortia focus on the actual business benefits that the solution provides - usually digitized and more efficient processes, which result in cost savings. The fact that DLT is used for this purpose is not specially emphasized, unless information technology executives are addressed directly. Some consortia still mention it at the same level with other business benefits (Cardossier, MediLedger). However, as other researchers have noted

Table 2: Milestones of the selected consortia.

Name	Foundation	Instit.	Pilot	Launch
B3i Re	Oct-16	Mar-18	Jun-18	Jul-19
Bakong	-	-	Jul-19	Oct-20
Cardossier	Nov-16	Mar-19	Apr-19	Jun-20
Contour	Jul-17	Jan-20	May-19	Oct-20
covantis	Oct-18	Mar-20	-	Feb-21
DL Freight	-	-	Jan-19	Oct-19
MediLedger	Jan-17	-	Feb-19	Oct-19
TradeLens	Jan-17	Aug-18	Jan-17	Dec-18
we.trade	Jan-17	Apr-18	Jun-18	Oct 18

[4], incentives work only if they represent business value for potential members ("executives don't care about blockchain").

Legal Form. There is no clear single best choice for the legal form of a DLT consortium. Most cases choose for-profit entities, such as Corporation, Limited or as a subdivision of an existing legal entity. For example, it was decided to make the TradeLens operating company a fully owned subsidiary of Maersk to ease legal approvals in countries around the world [8]. However, the more frequent case is incorporation of a new company where consortium members become shareholders (we.trade, B3i, Contour). Cardossier marks the exception by founding a non-profit association tasked with maintaining the consortium platform.

Disintermediation. One of the most frequently cited benefits of DLT is disintermediation. However, recent research has questioned the degree to which DLT actually accomplishes the task of eliminating the platform operator, calling it the "disintermediation fallacy" [31]. Our results partially support this hypothesis. Every reviewed case relies on an external company for A) platform development, B) platform hosting or C) both of these. While for case B the involvement of the third party is limited to infrastructure, in the other case more significant trust is required in the platform developer (and operator in case C). In these cases it is worth questioning if DLT is actually needed to support the platform, since the external platform operator already has sufficient trust and control to invalidate the disintermediation argument.

Interoperability. In some industries, there are several competing DLT consortia which serve similar business needs. This is especially apparent for Global Trade, where multiple operational and emerging platforms serve similar needs (i.e. Bill of Lading, Letter of Credit). In turn, interoperability is becoming a more important concern for platform operators. For example, TradeLens and we.trade built a bridge for trusted data transfer between the two networks as part of a research project [32]. While interoperability is not a primary concern before launch, it should be considered in strategic plans [8]. Especially if there are other DLT networks fulfilling similar needs, interoperability pilots should be conducted. These may even result in mergers which benefit both participants, as was the case when we.trade and Batavia merged in October 2018. DLT platforms are about network effects [33], so greater reach benefits all participants.

By analyzing six dimensions we have now answered RQ1. The results established some commonalities: successful consortia focus on business benefits as incentive for participation, they rely on trusted providers for DLT operation, and favor interoperability initiatives in long-term plans. As for differences, platform choice, network size and legal form are quite heterogeneous. They mostly seem to depend on the specific requirements of the case study and its industry.

6. DLT Consortium Lifecycle

This Section is dedicated to answering ***RQ2****: Is there a common lifecycle for DLT consortia?*

As mentioned in Section 3, we iteratively develop a lifecycle based on a detailed review of all selected cases. Table 2 shows the four milestones Founding, Institutionalization, Pilot and Launch for each consortium:

- **Founding**: first consortium announcement
- **Institutionalization**: legal entity creation
- **Pilot**: last pilot experiment
- **Launch**: launch announcement

Based on these milestones, we structure the lifecycle along the Trial and Integration phases detailed in Section 2. The result is shown in Figure 2. Initially, the *Formation* and *Pilot* phase involve incremental risk taking and trial & error during the evaluation of different concepts and pilots. Once sufficient trust is established among consortium partners, consortia proceed to the *Launch* and *Expansion* phases. Each of these phases consists of multiple sub-phases, which are not necessarily sequential in order. The lifecycle is meant to depict the most common approach. If there is an exception to this approach, it is mentioned in the description of the phases hereafter.

6.1. Formation

Any DLT consortium is at some point initiated by a single organization reaching out to others for collaboration. This organization is hereafter referred to as consortium *initiator*. The initiator often becomes the leading driver (*leader*) of the consortium (Bakong, TradeLens, DL Freight). In other cases, the leader is a newly formed entity determined by consortium shareholders (Cardossier, covantis, we.trade).

Prototyping. The consortium initiator initially develops a concept and vision for the platform. This vision is subject to change during later collaboration with other participants, but it encompasses the basic foundation of the collaboration. This idea is often related to exchanging business documents digitally based on DLT, to improve auditability and traceability. This phase may include proof of concepts

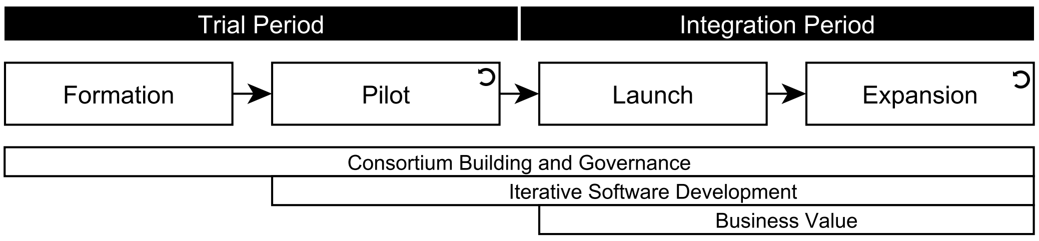


Figure 2: DLT consortium lifecycle phases.

Table 3: Phases and sub-phases of the consortium lifecycle.

Phase	Sub-phases	Description
Formation	Prototyping	Ideation and first software prototype concepts
	Partner Search	Initiator searches for consortium and implementation partners
	Institutionalization	A legal entity is formed to represent the consortium (not applicable for pre-existing partner networks)
Pilot	Requirements Engineering	Requirements and an initial scope for the business case are established. This includes compliance and security-based requirements
	Development and Testing	Prototype software is developed by the consortium institution and/or software provider
	Process experiment	Prototype is used in an actual business process for testing purposes
Launch	Production Use	First product is launched for production use
	Commercialization	Revenue model developed and consortium institution begins earning revenue (only for-profit consortia)
	Business Value	Participants are seeing first benefits as a result of adoption
Expansion	Ecosystem Building	More peripheral organizations in the supply chain join the platform (often as users, not operators)
	Expansion of Scope	Additional business cases (products) within the consortium are developed and launched

to demonstrate the feasibility and potential benefits of a collaboration among the initial partners.

Partner Search. The consortium lead focuses on acquiring suitable partners for the use case. Convincing partners can be the most difficult part of building a consortium, especially if these partners are competitors. An integral part of the trial period is building trust in partners, especially in the consortium lead and that it will be able to navigate ecosystem tensions [8]. During this phase, it can be helpful to focus on first acquiring partners that are not direct competitors, which builds trust initially [7]. While the partner search activity begins in the formation phase, it continues throughout all later phases as the consortium continually seeks to grow its reach. This is fact is represented in Figure 2 as *Consortium Building*.

Institutionalization. Eventually, newly formed consortia are confronted with the need for a legal entity. The Cardossier project identified several reasons related to obtaining a critical mass and network effects, as well as complying with laws and regulation [7]. As mentioned previously in Section 5, the specific legal form varies. The branding often changes along with institutionalization. For example, we.trade rebranded from its original name Digital Trade Consortium, Contour from Voltron, and TradeLens received its current name on commercialization after multiple renamings [8]. Previously existing business networks (Bakong) or networks initiated and led by a software service provider (MediLedger, DL Freight) do not face this need, as the consortium initiator takes on the role of consortium representation. While Institutionalization is part of the Formation phase, it often takes place during the Pilot phase or even during the Launch phase (before the launch date). This can be attributed to the fact that establishing a joint entity as shareholders requires some trust building.

6.2. Pilot

The *Pilot* phase begins with development of the pilot software and finishes with the end of the last process experiment. A pilot launch refers to a limited roll-out of the target platform. In most cases the prototype used for this purpose is feature-complete, and some projects refer to it as the Minimum Viable Product (MVP). During the *Process Experiment* it is made available to a small set of test partners. Depending on the success and scope of the pilot, multiple pilots may be needed before the project consortium transitions to *Launch*.

Requirements engineering. The business case is finalized based on the results of *Prototyping* during the *Formation* phase. Functional and non-functional requirements of the target DLT platform are determined. To this end, consortium partners collaborate with a software service provider. Requirements are usually based on business and regulatory concerns. Requirements that are common to several reviewed DLT consortia include scalability, privacy and interoperability. MediLedger collaborated with a regulator (the US Food & Drug Administration) to ensure compliance of the pilot with regulation [9].

Development and Testing. For almost all cases, a single company is tasked with software development in this phase of the project. In some consortia, the newly formed legal entity hires specialized employees itself (Contour). In other cases, the consortium initiator is a software service provider and thus also develops the platform (Cardossier, DL Freight, MediLedger). Finally, others hire service providers specialized for DLT development (TradeLens, we.trade, covantis). Commonly, such service providers also provide a cloud-based platform where node operators can control DLT nodes. This avoids the need to train people in each participating organization on how to set up a DLT node. Another important decision that occurs during this phase is the choice of DLT platform. The initial choice is not final, as pilot results may lead to the decision to switch to another framework better suited to the business case. This phase also marks the beginning of the *Iterative Software Development* process [23] marked in Figure 2 that continues throughout later phases.

Process experiment. Finally, an experiment is conducted within the target operational business process. This experiment is of limited scope, for example by concerning a specific product in supply chain cases. For one pilot experiment, TradeLens conducted its pilot experiment with the roses supply chain and involved only necessary partners [8]. Trade Finance consortia start with one or multiple pilot financing transactions between banks and businesses. As one example, Contour performed a total of 10 pilot experiments with different partners over the course of 1 year [27].

6.3. Launch

When the consortium is satisfied with pilot results, it usually proceeds to launch the full platform for all consortium participants. While the launch itself is tied to a single point in time (see Table 2), it is accompanied by many preparations and post-launch effects, which are addressed in this phase.

Production Use. In practice, members rarely use the platform for all business transactions immediately after the launch date. In most cases, members finish platform onboarding months after the launch announcement. Often there are separate announcements when a member has completed onboarding [8] or their first successful transaction [27]. These delays can be explained by varying commitment levels of the partners, but most importantly by challenges in integrating novel DLT with existing business processes and legacy systems.

Commercialization. First revenue streams are beginning to materialize for the consortium's legal entity. During the first months after launch, platform usage is usually low, as members begin to move more and more transactions to the new system [25]. Revenue models are usually based on recurring membership fees for consortium members [29, 7, 9]. Notably, TradeLens offers free access to some ecosystem members like container terminals and authorities, while others must pay a subscription and transaction fees [8].

Business Value. Consortium members notice first positive returns on their investment. These are use case dependent, but principally include lower delays (trade finance cases), increased supply chain transparency (product tracking cases) and improved process efficiency through digitization of paper-based documents. For example, DL Freight reduced invoice disputes, which lowered accounts receivable for carriers and costs for Walmart [29]. Others like Bakong and B3i cite significant cost and time savings [26, 25].

6.4. Expansion

During the *Expansion* phase, the consortium focuses on increasing business value. This phase begins about 6 - 12 months after the platform has been successfully adopted, deployed and used at all members.

Governance. During expansion of the network, new decisions require consensus building among consortium members. These concern

- new member admission
- new feature prioritization
- accommodating regulatory concerns
- platform monetization
- software updates
- handling security incidents

While some rules are part of the initial consortium agreements, these are ongoing concerns that may require intervention during operation. For example, DL Freight requires chaincode software updates to be approved by a majority of participants [29]. In addition, security concerns exist and may be hard to deal with in a cooperative environment [15].

Ecosystem Building. To ease member acquisition, consortia focus on building an ecosystem around the core DLT platform after launch. While most platforms include the most important core features initially, additional integrations with existing systems ease member acquisition. Another focus during this phase is onboarding related actors that need access to trustworthy DLT data, but don't necessarily validate transactions on the platform. These include auditors, government authorities and suppliers/customers of consortium members.

Expansion of Scope. Gradually, consortia agree on adding new features and products to the platform to expand its scope and concomitant business benefits. In addition to incremental improvements, consortia often bundle major features or new products in a major version release. For example, DL Freight and B3i have announced 2.0 versions of their platforms with significantly expanded capabilities.

7. Discussion

Democratic governance vs. benevolent dictators. Democratic decisions are important for consortium longevity and trust-building. For example, for TradeLens the decision-making changed from the platform owner making all the decisions to a more democratic model governed by an advisory board [8]. While some consortia favor democratic decisions, others prefer a "benevolent dictator" approach [9]. In DL Freight, while chaincode updates require approval by carriers, the core direction of the platform is determined by Walmart Canada and DL Freight [29]. Cardossier followed an hierarchical off-chain governance model initially, but plans to transition to a more democratic model in the long-term [7]. To summarize, initially a more hierarchical approach can help consortia move quickly toward *Pilot* and *Launch*, but during the *Expansion* phase democratic governance is preferable to ensure stakeholder expectations are met.

Disintermediation. Both theory and practice have long claimed that DLT disintermediates trusted third parties (TTPs) [4, 31]. However, all consortia in our study agree on the fact that an independent entity is needed to coordinate the consortium's technology development. While this may seem counterintuitive at first, it is a logical consequence of coopetition in partner networks. Practitioners have noted that the need for competitors to cooperate is one of the hardest challenges to solve, often requiring a sponsor such as an industrial body to step in [5]. For some consortia like Bakong, there is a natural sponsor (NBC). For others, new institutions are created for this purpose as TTPs, which are controlled by the members as shareholders. Whether the TTP's software platform must be DLT-based remains controversial, as consortia like Komgo[6] have transitioned away from DLT entirely in favor of a centralized database. As the blockchain/DLT buzzword slowly loses its appeal (cf. *Incentives* dimension in Section 5), the focus is now on realizing actual business benefits of DLT.

Study limitations. Three consortia were not included since they failed to meet Q2, so they may be included in future studies. Several other candidates were not included due to Q1, a lack of information available on the development phases. Additionally, there is a slight bias towards western use cases, as many Asian consortia (especially in China) are inaccessible to the authors due to language barriers.

8. Conclusion

We performed a multiple case comparative study to find commonalities and differences among successful DLT consortia. Following a strict selection methodology based on contextual and intellectual information quality, 9 successful consortia were selected. The findings were structured along 6 dimensions and used to develop a new lifecycle theory for DLT consortia. They challenge the disintermediation aspect of permissioned DLTs and open up an avenue for future research. They also provide insights for early stage DLT consortia, as well as informing future DLT case studies and DLT platform theory.

Acknowledgment. We would like to thank our interview partners for their valued time and input toward improving data accuracy and the lifecycle.

[6]`www.komgo.io`

References

[1] S. Tönnissen and F. Teuteberg, "Analysing the impact of blockchain-technology for operations and supply chain management: An explanatory model drawn from multiple case studies," *International Journal of Information Management*, 2020.

[2] R. Ziolkowski, G. Miscione, and G. Schwabe, "Decision Problems in Blockchain Governance: Old Wine in New Bottles or Walking in Someone Else's Shoes?," *Journal of Management Information Systems*, 2020.

[3] S. Meiklejohn, "Top Ten Obstacles along Distributed Ledgers Path to Adoption," *IEEE Security Privacy*, vol. 16, pp. 13–19, July 2018.

[4] M. Lacity and R. Van Hoek, "What we've learned so far about blockchain for business," *MIT Sloan Management Review*, vol. 62, no. 3, pp. 48–54, 2021.

[5] B. Carson, G. Romanelli, P. Walsh, and A. Zhumaev, "Blockchain beyond the hype: What is the strategic business value," *McKinsey & Company*, pp. 1–13, 2018.

[6] M. Budman, R. Bhat, and S. Bordoloi, "Deloitte's 2020 Global Blockchain Survey - From promise to reality," 2020.

[7] L. Zavolokina, R. Ziolkowski, I. Bauer, and G. Schwabe, "Management, governance, and value creation in a blockchain consortium," *MIS Quarterly Executive*, 2020.

[8] T. Jensen, J. Hedman, and S. Henningsson, "How TradeLens delivers business value with blockchain technology," *MIS Quarterly Executive*, 2019.

[9] J. Mattke, A. Hund, C. Maier, and T. Weitzel, "How an enterprise blockchain application in the U.S. Pharmaceuticals supply chain is saving lives," *MIS Quarterly Executive*, 2019.

[10] R. K. Yin, *Case Study Research: Design and Methods*. 2009.

[11] M. Hearn, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, 2016.

[12] A. Larson, "Partner networks: Leveraging external ties to improve entrepreneurial performance," *Journal of business venturing*, vol. 6, no. 3, pp. 173–188, 1991.

[13] R. Lapiedra, S. Smithson, J. Alegre, and R. Chiva, "Role of information systems on the business network formation process: An empirical analysis of the automotive sector," *Journal of Enterprise Information Management*, vol. 17, pp. 219–228, Jan. 2004.

[14] A. B. Pedersen, M. Risius, and R. Beck, "A ten-step decision path to determine when to use blockchain technologies," *MIS Quarterly Executive*, vol. 18, no. 2, pp. 99–115, 2019.

[15] B. Putz and G. Pernul, "Trust Factors and Insider Threats in Permissioned Distributed Ledgers," *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, vol. XLII, pp. 25–50, 2019.

[16] R. Ziolkowski, G. Parangi, G. Miscione, and G. Schwabe, "Examining gentle rivalry: Decision-making in blockchain systems," in *52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019* (T. Bui, ed.), pp. 1–10, ScholarSpace, 2019.

[17] R. van Pelt, S. Jansen, D. Baars, and S. Overbeek, "Defining Blockchain Governance: {A} Framework for Analysis and Comparison," *Inf. Syst. Manag.*, vol. 38, no. 1, pp. 21–41, 2021.

[18] R. Beck, C. Müller-Bloch, and J. L. King, "Governance in the blockchain economy: A framework and research agenda," *Journal of the Association for Information Systems*, 2018.

[19] M. Rauchs, A. Blandin, K. Bear, and S. B. McKeon, "2nd Global Enterprise blockchain benchmarking study," *Available at SSRN 3461765*, 2019.

[20] M. O. Deventer, F. Berkers, M. Vos, A. Zandee, T. Vreuls, L. van Piggelen, A. Blom, B. Heeringa, S. Akdim, P. van Helvoort, *et al.*, "Techruption Consortium Blockchain: What it takes to run a blockchain together," in *Proceedings of 1st ERCIM Blockchain Workshop 2018, Amsterdam, Netherlands 8-9 May 2018*, European Society for Socially Embedded Technologies (EUSSET), 2018.

[21] M. del Castillo, "Blockchain 50 2021." `https://www.forbes.com/sites/michaeldelcastillo/2021/02/02/blockchain-50/`, Feb. 2021.

[22] E. Ganne and D. Patel, "Blockchain & DLT in Trade: Where do we stand?," 2020.

[23] I. Jacobson, G. Booch, and J. E. Rumbaugh, *The Unified Software Development Process - the Complete Guide to the Unified Process from the Original Designers*. Addison-Wesley Object Technology Series, Addison-Wesley, 1999.

[24] R. Y. Wang and D. M. Strong, "Beyond Accuracy: What Data Quality Means to Data Consumers," *Journal of Management Information Systems*, vol. 12, pp. 5–33, Mar. 1996.

[25] R3, "R3 B3i Case Study," Aug. 2020.

[26] National Bank of Cambodia, "Project Bakong - Next Generation Payment System," tech. rep., Oct. 2020.

[27] R3, "R3 Contour Case Study," June 2020.

[28] ConsenSys, "Covantis Case Study: Modernizing Global Supply Chains with ConsenSys Blockchain Solutions." `https://consensys.net/blockchain-use-cases/global-trade-and-commerce/covantis/`, Oct. 2021.

[29] M. Lacity and R. Van Hoek, "Requiem for reconciliations: DL Freight, a blockchain-enabled solution by Walmart Canada and DLT Labs," 2021.

[30] IBM, "We.trade Case Study." `https://www.ibm.com/case-studies/wetrade-blockchain-fintech-trade-finance`, 2018.

[31] F. Hawlitschek, B. Notheisen, and T. Teubner, "A 2020 perspective on "The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy"," *Electron. Commer. Res. Appl.*, vol. 40, p. 100935, 2020.

[32] E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, and C. Vecchiola, "Enabling Enterprise Blockchain Interoperability with Trusted Data Transfer (Industry Track)," in *Proceedings of the 20th International Middleware Conference Industrial Track*, (New York, NY, USA), pp. 29–35, Association for Computing Machinery, 2019.

[33] D. Roeck, H. Sternberg, and E. Hofmann, "Distributed ledger technology in supply chains: A transaction cost perspective," *International Journal of Production Research*, vol. 58, no. 7, pp. 2124–2141, 2020.

Benedikt Putz

Department of Information Systems
Faculty of Business, Economics, and
Management Information Systems
University of Regensburg
Universitätsstr. 31, 93053 Regensburg

updated: August, 2022
ORCID: 0000-0002-4265-1106
email: benedikt.putz@ur.de
research group: www-ifs.ur.de
website: go.ur.de/putz

Education

2019 - 2022	PhD Student *University of Regensburg, Germany*
2016 - 2018	M.Sc. Management Information Systems (Honors) *University of Regensburg, Germany*
2017	Exchange Program — Information Processing Science *University of Oulu, Finland*
2012 - 2015	B.Sc. Business Information Systems *University of Augsburg, Germany*
2009 - 2010	CBYX/PPP Youth Exchange *Bedford High School, PA, USA*

Reviewing Activities

Conferences. ARES 2022, WI 2022, COMPSAC 2022, ESORICS 2022, CAiSE 2022, WPES 2021, DBSec 2021, ARES 2021, WISE 2021, WI 2021, NSS 2020, SPBP 2020, ESORICS 2020, ICCCN 2020, SECRYPT 2020, CAiSE 2020, ICISSP 2020, BPM 2019, ESORICS 2019, SECRYPT 2019, WISE 2019, DBSec 2019, ICISSP 2019

Journals. Future Generation Computer Systems, International Journal of Information Security (2x), Journal of Cybersecurity and Privacy

Third-party projects

TRIO. Responsible for transferring information security research into practice as part of the TRIO project for fostering transfer and innovation in Eastern Bavaria (2018-2022).

DEFENSIVE. Wrote a successful project proposal for a decentralized data escrow platform for threat intelligence sharing as a 3 year project funded by BMBF.

Publications

Benedikt Putz and Günther Pernul. 2022. Comparing Successful DLT Consortia: A Lifecycle Perspective. In 55th Hawaii International Conference on System Sciences 2022, 4591–4600.

Benedikt Putz, Fabian Böhm, and Günther Pernul. 2021. HyperSec: Visual analytics for blockchain security monitoring. In ICT systems security and privacy protection - 36th IFIP TC 11 International Conference, SEC 2021, Oslo, Norway, June 22-24, 2021, Proceedings (IFIP Advances in Information and Communication Technology), Springer, 165–180.

Benedikt Putz, Marietheres Dietz, Philip Empl, and Günther Pernul. 2021. Ethertwin: Blockchain-based secure digital twin information management. Information Processing & Management 58, 1 (2021).

Benedikt Putz and Günther Pernul. 2020. Detecting Blockchain Security Threats. In 2020 IEEE International Conference on Blockchain (Blockchain), IEEE, 313–320.

Rafael Belchior, Benedikt Putz, Günther Pernul, Miguel Correia, André Vasconcelos, and Sérgio Guerreiro. 2020. SSIBAC: Self-Sovereign Identity Based Access Control. In The 3rd International Workshop on Blockchain Systems and Applications (BlockchainSys2020), in Conjunction with IEEE TrustCom 2020, IEEE.

Florian Menges, Benedikt Putz, and Günther Pernul. 2020. DEALER: decentralized incentives for threat intelligence reporting and exchange. International Journal of Information Security (2020).

Marietheres Dietz, Benedikt Putz, and Günther Pernul. 2019. A Distributed Ledger Approach to Digital Twin Secure Data Sharing. In Data and Applications Security and Privacy XXXIII, Simon N. Foley (ed.). Springer International Publishing, 281–300.

Benedikt Putz and Günther Pernul. 2019. Trust Factors and Insider Threats in Permissioned Distributed Ledgers. Transactions on Large-Scale Data- and Knowledge-Centered Systems XLII, (2019), 25–50.

Benedikt Putz, Florian Menges, and Günther Pernul. 2019. A secure and auditable logging infrastructure based on a permissioned blockchain. Computers & Security 87, (November 2019), 101602.

TALKS

2022	Comparing Successful DLT Consortia: A Lifecycle Perspective *Hawaii International Conference on System Sciences 2022*
2021	HyperSec: Visual analytics for blockchain security monitoring *IFIP Sec 2021*
2021	Hyperledger Fabric security monitoring based on Hyperledger Explorer *Hyperledger Global Forum 2021*
2020	Detecting Blockchain Security Threats *2020 IEEE International Conference on Blockchain*