RailChain: Abschlussbericht

Ingo Schwarzer, Said Weiß-Saoumi, Roland Kittel, Tobias Friedrich, Koraltan Kaynak, Cemil Durak, Andreas Isbarn, Jörg Diestel, Jens Knittel, Marquart Franz, Carlos Morra, Susanne Stahnke, Jens Braband, Johannes Dittmann, Stephan Griebel, Andreas Krampf, Martin Link, Matthias Müller, Jens Radestock, Leo Strub, Kai Bleeke, Leander Jehl, Rüdiger Kapitza, Ines Messadi, Stefan Schmidt, Signe Schwarz-Rüsch, Lukas Pirl, Robert Schmid, Dirk Friedenberger, Jossekin Beilharz, Arne Boockmeyer, Andreas Polze, Ralf Röhrig, Hendrik Schäbe, Ricky Thiermann

Technische Berichte Nr. 152

des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Universität Potsdam

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 152

Ingo Schwarzer | Said Weiß-Saoumi | Roland Kittel | Tobias Friedrich |
Koraltan Kaynak | Cemil Durak | Andreas Isbarn | Jörg Diestel | Jens Knittel |
Marquart Franz | Carlos Morra | Susanne Stahnke | Jens Braband |
Johannes Dittmann | Stephan Griebel | Andreas Krampf | Martin Link |
Matthias Müller | Jens Radestock | Leo Strub | Kai Bleeke | Leander Jehl |
Rüdiger Kapitza | Ines Messadi | Stefan Schmidt | Signe Schwarz-Rüsch |
Lukas Pirl | Robert Schmid | Dirk Friedenberger | Jossekin Beilharz |
Arne Boockmeyer | Andreas Polze | Ralf Röhrig | Hendrik Schäbe |
Ricky Thiermann

RailChain

Abschlussbericht

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar

Universitätsverlag Potsdam 2023
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Layout: Tobias Pape
Druck: docupoint GmbH Magdeburg

ISBN 978-3-86956-550-7

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
https://doi.org/10.25932/publishup-57740
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-577409

Preface

The *RailChain* project designed, implemented, and experimentally evaluated a juridical recorder that is based on a distributed consensus protocol. That juridical blockchain recorder has been realized as distributed ledger on board the *advanced TrainLab* (ICE-TD 605 017) of Deutsche Bahn.

For the project, a consortium consisting of DB Systel, Siemens, Siemens Mobility, the Hasso Plattner Institute for Digital Engineering, Technische Universität Braunschweig, TÜV Rheinland InterTraffic, and Spherity has been formed. These partners not only concentrated competencies in railway operation, computer science, regulation, and approval, but also combined experiences from industry, research from academia, and enthusiasm from startups.

Distributed ledger technologies (DLTs) define distributed databases and express a digital protocol for transactions between business partners without the need for a trusted intermediary. The implementation of a blockchain with real-time requirements for the local network of a railway system (e.g., interlocking or train) allows to log data in the distributed system verifiably in real-time. For this, railway-specific assumptions can be leveraged to make modifications to standard blockchains protocols.

EULYNX and OCORA (Open CCS On-board Reference Architecture) are parts of a future European reference architecture for control command and signalling (CCS, Reference CCS Architecture – RCA). Both architectural concepts outline heterogeneous IT systems with components from multiple manufacturers. Such systems introduce novel challenges for the approved and safety-relevant CCS of railways which were considered neither for road-side nor for on-board systems so far. Logging implementations, such as the common juridical recorder on vehicles, can no longer be realized as a central component of a single manufacturer. All centralized approaches are in question.

The research project *RailChain* is funded by the *mFUND* program and gives practical evidence that distributed consensus protocols are a proper means to immutably (for legal purposes) store state information of many system components from multiple manufacturers. The results of *RailChain* have been published, prototypically implemented, and experimentally evaluated in large-scale field tests on the *advanced TrainLab*. At the same time, the project showed how *RailChain* can be integrated into the road-side and on-board architecture given by OCORA and EULYNX.

Logged data can now be analysed sooner and also their trustworthiness is being increased. This enables, e.g., auditable predictive maintenance, because it is ensured that data is authentic and unmodified at any point in time.

Vorwort

Das Projekt *RailChain* hat einen verteilten *Juridical Recorder* entworfen, implementiert und experimentell evaluiert, der auf einem echtzeitfähigen verteilten Konsensprotokoll basiert. Dieser *Juridical Blockchain Recorder* wurde als *distributed ledger* an Bord des *advanced TrainLab*s der Deutschen Bahn (ICE-TD 605 017) umgesetzt.

Für das Projekt hat sich ein Konsortium aus DB Systel, Siemens, Siemens Mobility, dem Hasso-Plattner-Institut für Digital Engineering, der Technischen Universität Braunschweig, sowie TÜV Rheinland InterTraffic und Spherity formiert und dabei Kompetenzen aus den Bereichen Bahnbetrieb, Informatik und Zulassungswesen gebündelt. Die Partner kombinieren Erfahrungen aus der Industrie und die akademische Forschung mit der Aufbruchstimmung aus dem Start-Up-Umfeld.

Distributed-Ledger-Technologien (DLTs) definieren verteilte Datenbanken und stellen ein digitales Protokoll für Transaktionen zwischen Geschäftspartnern dar, ohne dass ein Mittelsmann beteiligt sein müsste. Die Implementierung einer Blockchain mit Echtzeitanforderungen für das lokale Netzwerk einer Eisenbahnanlage (z. B. Stellwerk oder Zug) erlaubt es, die im verteilten System entstehenden Daten nachweislich in Echtzeit zu protokollieren. Dabei können eisenbahnspezifische Randbedingungen ausgenutzt werden, um Standard-Blockchain-Protokolle anzupassen.

EULYNX und OCORA (*Open CCS On-board Reference Architecture*) sind Bestandteile einer zukünftigen europäischen Referenzarchitektur für das Leit- und Sicherungssystem (*Reference CCS Architecture* – RCA, *Control Command and Signalling* – CCS). Beide Architekturkonzepte skizzieren herstellerübergreifende, komponentenbasierende heterogene IT-Systeme. Solche Systeme bergen neue Herausforderungen, die bislang im Kontext der zugelassenen, sicherheitsrelevanten Leit- und Sicherungstechnik der Bahn weder strecken- noch fahrzeugseitig adressiert werden mussten. Logbuch-Implementierungen, wie der gängige *Juridical Recorder* auf Fahrzeugen, können nun nicht mehr als zentrale Systemkomponente eines einzelnen Herstellers umgesetzt werden. Alle zentralisierten Lösungsansätze sind in Frage gestellt.

Das *mFUND*-geförderte Forschungsprojekt erbringt den praktischen Nachweis, dass Zustandsinformationen über eine Vielzahl von Systemkomponenten herstellerübergreifend und gerichtsfest mittels verteilten Konsensprotokollen gespeichert werden können. Ergebnisse von *RailChain* wurden publiziert, prototypisch implementiert und in großen Feldtests auf dem *advanced TrainLab* experimentell evaluiert. Gleichzeitig wurde aufgezeigt, wie sich *RailChain* in den mit OCORA und EULYNX vorgegebenen fahrzeug- und streckenseitigen Architekturentwurf integrieren lässt.

Daten können dadurch zeitnaher ausgewertet werden und gleichzeitig wird ihre Vertrauenswürdigkeit erhöht. Dies ermöglicht u. a. nachvollziehbare zustandsorientierte Wartung, denn es kann jederzeit sichergestellt werden, dass die Daten authentisch sind und auch nicht verändert wurden.

Inhaltsverzeichnis

1 Einleitung

Das Bundesministerium für Digitales und Verkehr (BMDV) hat den Modernitätsfonds *mFUND* [4] initiiert. Ein Leitgedanke des Förderprogramms besteht darin, allen interessierten Akteuren – im Sinne eines Ansatzes *Open Data* – breiten Zugang zu den Daten des BMDV und seines Geschäftsbereichs zu gewähren und damit Innovationen und umsetzungsnahe Anwendungsfälle für die Datennutzung zu ermöglichen. Ziel des Forschungsprogramms ist es, auf Basis der vom BMDV bereitgestellten Daten innovative Lösungsansätze und Anwendungen zu entwickeln und so die ökonomischen und gesellschaftlichen Potenziale für moderne Anwendungen „von Big Data zu Smart Data“ zu nutzen. Durch diese Nutzbarmachung sollen digital verfügbare Daten die „Ressource“ für den Fortschritt und für die Stärkung des digitalen Standorts Deutschland darstellen und dazu beitragen, tägliche Mobilitätsszenarien wirtschaftlicher, komfortabler, sicherer und umweltfreundlicher zu machen. Das Förderprogramm leistet damit auch ein Beitrag zur Förderung einer neuen Datenkultur, die Big Data als Chance begreift.

Distributed-Ledger-Technologien (DLTs) inkl. der darauf aufbauenden Blockchain-Technologie definieren verteilte Datenbanken und stellen ein digitales Protokoll für Transaktionen zwischen Geschäftspartnern dar, ohne dass ein Mittelsmann wie etwa eine Bank oder ein Bezahlsystem wie PayPal beteiligt sein müsste. Überall dort, wo der Transfer von Daten oder Werten erfasst werden muss, könnte die DLT z. B. durch computerüberwachte Verträge (sog. *Smart Contracts*) Prozesse vereinfachen, automatisieren und sicherer machen. Drei mathematische Fachrichtungen sind hier auf neue und elegante Weise verknüpft: Kryptografie, verteilte Systeme und Spieltheorie. In einer Blockchain ist die zeitliche Abfolge von Transaktionen exakt und jederzeit öffentlich nachvollziehbar. Im öffentlichen Fokus stehen derzeit vor allem finanzielle und rechtliche Transaktionen, wo die Vorteile von dezentralen Datenbanken mit *Peer-to-Peer*-Ansätzen über Unternehmensgrenzen hinweg zum Tragen kommen. Für die Siemens AG und ihre verbundenen Unternehmen hat das in vielerlei Hinsicht große Bedeutung, etwa beim Stromhandel in komplexen Märkten oder bei Dienstleistungen auf digitaler Basis. Blockchain wird daher als eine der zukünftigen Kerntechnologien des Konzerns eingeschätzt.

Die Implementierung einer Blockchain für das lokale Netzwerk einer Eisenbahnanlage (z. B. Stellwerk oder Zug) erlaubt es, im verteilten System entstehende Daten nachweislich in Echtzeit zu protokollieren. Dabei können eisenbahnspezifische Randbedingungen ausgenutzt werden, um anwendungsspezifische Anpassungen an Standard-Blockchains vorzunehmen (z. B. *Proof of Authority*, *Proof of Kernel Work* oder *Proof of Stake* anstelle von *Proof of Work*).

Das Ziel des Vorhabens war in erster Linie die Spezifikation und Implementierung eines Blockchain-Demonstrators für das Eisenbahnwesen, der eine Basisfunk-

tionalität umsetzt, auf der Echtzeitanwendungen (< 1 Sekunde Block-Zyklus-Zeit) ablaufen können, sowie dessen Erprobung in einem geeigneten Testfeld.

Dadurch konnte z. B. ein dezidierter *Juridical Recorder* eingespart bzw. ersetzt werden. Durch weitgehenden Einsatz bzw. Anpassung von Standardkomponenten wird weiterhin die Effektivität der Eisenbahn erhöht. Neben der technischen Übertragbarkeit in die Bahntechnik soll auch die Wirtschaftlichkeit gezeigt und in Use Cases weiter konkretisiert werden.

2 Stand der Wissenschaft und Forschung

Verteilte Systeme sind seit jeher von hohem Interesse für die Computerwissenschaften. Dementsprechend sind viele Fragestellungen und Erkenntnisse lange bekannt, wie beispielsweise Fehlerklassen bei verteilter Konsensfindung (Lamport et al. 1982, Byzantine generals problem) oder Grundlagen der Blockchain-Technologie (Haber et al. 1990, „How to time-stamp a digital document"). Durch immer neue betrachtete Aspekte und immer neue Anforderungsfälle, sind verteilte Systeme im Allgemeinen und Blockchains im Speziellen jedoch noch immer Gegenstand der Forschung. Gerade die Berücksichtigung von domänenspezifischen Randbedingungen und die Ableitung von Annahmen für spezifische Anwendungen eröffnet Raum für zunächst scheinbar unrealistische Lösungen. Im Kontext von *RailChain* sind somit die wenigen Arbeiten zu den Themen *Distributed-Ledger*-Technologien (DLTs) und Blockchain im Eisenbahnbereich, sowie für juristische Datenaufzeichnung verwandt.

Blockchain in Eisenbahnsystemen Kuperberg et al. [42, 43] bewerten *Smart Contracts* zur Dezentralisierung des Eisenbahnbetriebs. Blockchains im Schienenverkehr wurden untersucht, um Zugverspätungen zu analysieren [44], für die Authentifizierung von Knoten in *Wireless Sensor Networks* [55] und zur Verbesserung der digitalen Fahrkartenausstellung [48]. Die Arbeiten [45, 47] konzentrieren sich auf digitales Ticketing, Lieferkettenmanagement und Verteilung von Daten. Unser System ist unseres Wissens nach das erste, das Blockchains im operativen Eisenbahnbetrieb einsetzt.

Juridical Recording Hartong et al. [28] sichern Daten der *Juridical Recording Unit* (JRU) durch den Austausch von Geheimnissen zwischen mehreren Parteien. Auf Blockchain-Technologien basierende Datenschreiber wurden für autonome Fahrzeuge [7, 26, 30, 46, 51], Roboter [53] und den Bereich des *Internet of Things* (IoT) [52] vorgeschlagen. Sie verwenden entweder BFT (Byzantinische Fehlertoleranz) zwischen mehreren Fahrzeugen [7, 30, 46], entwickeln neue Ansätze zur Einigung [26] oder stützen sich auf komplexe Blockchains [51, 52]. ZugChain [50] – die Implementierung des Projekts *RailChain* – sichert die Übereinstimmung von juristischen Daten die in einem Zug protokolliert werden müssen. Dafür wird eine leichtgewichtige Blockchain-Infrastruktur genutzt. Hinsichtlich des begrenzten Speicherplatzes präsentieren Wang et al. [52] eine hierarchische Blockchain, die einen Export unter Verwendung einer Runde eines byzantinisch fehlertoleranten Konsens durchführt. Unser Export ist unabhängig von der Vereinbarung, um den Overhead zu reduzieren und die Protokollierung nicht zu beeinträchtigen.

3 Beschreibung der Use Cases

Aufgrund der Relevanz bei der Analyse von Unfällen, besitzt die Aufzeichnung und Speicherung von Fahrdaten in Eisenbahnanwendungen einen hohen Stellenwert. Da die Daten juristisch relevant sind, ist deren Integrität besonders wichtig. Auf Zügen übernimmt die *Juridical Recording Unit* (JRU) – auch als Fahrdatenschreiber oder umgangssprachlich als *Black Box* bezeichnet – die Aufzeichnungen der Fahrdaten. Stand der Technik für eine JRU ist ein *On-board*-Gerät mit unfallsicherem Speichermedium, das in Lokomotiven und Triebzügen verbaut wird und verschiedene Schnittstellen zur Daten- bzw. Signalaufnahme besitzt. Um die aufgezeichneten Daten zu entnehmen werden bisher kabelgebundene Verfahren eingesetzt. Die physische Beschränkung des Zugangs zu der JRU und ein in der Anwendung implementiertes Rechtesystem sichern die Integrität der Daten und personenbezogene Informationen.

Blockchain- bzw. *Distributed-Ledger*-Technologien (DLTs) bieten insbesondere bei der Aufzeichnung relevanter Daten mit hohen Anforderungen an Integrität in verteilten Infrastrukturen, wie es Eisenbahnanwendungen sind, ein hohes Potenzial. Durch die räumlich verteilte Speicherung der Fahrdaten auf verschiedenen Rechnern kann die Sicherheit gegenüber Verlust der aufgezeichneten Daten erhöht werden. Blockchain-Technologien stellen durch Konsensmechanismen und den Einsatz starker Kryptographie sicher, dass die Daten über die beteiligten Systeme hinweg konsistent sind und vor Manipulation geschützt sind. Insbesondere vor dem Hintergrund der fortschreitenden Fragmentierung und Privatisierung der Eisenbahnindustrie, bei denen das Vertrauen zwischen allen Parteien zukünftig nicht notwendigerweise vorhanden ist, bieten diese Technologien bedeutendes Potenzial.

Das Projekt *RailChain* hat sich zum Ziel gesetzt, die Machbarkeit und den Nutzen der DLT bei der Aufzeichnung von technischen Informationen zu erforschen und wesentliche Grundlagen und Voraussetzungen zu ermitteln, um einem praktischen Einsatz den Weg zu ebnen.

Die Anforderungen für das Projekt sind in drei Use Cases verfasst. Diese Use Cases fokussieren sich auf unterschiedliche Teilaspekte und bauen aufeinander auf. Bei der Formulierung der Anforderungen wird berücksichtigt, dass die Erkenntnisse des Projektes sowohl im Labormaßstab als auch im Probebetrieb auf dem *advanced TrainLab* gewonnen werden. Somit basieren die Ergebnisse letztendlich auf praktischen Erfahrungen.

Use Case 1: *Asset Identity* inkl. *Asset Tracking* Jede Informationsquelle kann mit einer eigenen digitalen Identität (properAsset Identity) ausgestattet werden. Das eröffnet weitreichende Möglichkeiten der eindeutigen Zuordnung von einzelnen Bauteilen, der Sicherheit über die ordnungsgemäße Zusammensetzung des Zuges

und der automatisierten Erfassung des gesamten Lebenszyklus dieser Bauteile (Asset Tracking). Der Use Case 1 definiert die Anforderungen für das *RailChain*-Projekt, um die Voraussetzungen für eine flächendeckend einzusetzende *Asset Identity* zu erforschen und den Nutzen darzustellen.

Use Case 2: Daten-Logger ohne Echtzeitanforderungen Dieser Use Case beschreibt die Anforderungen an das Projekt die DLT für Aufzeichnungsfunktionen einzusetzen. Dabei steht im Vordergrund, die Rahmenbedingungen für den sinnvollen Einsatz der Technologie zu ermitteln und ggf. zu demonstrieren.

Use Case 3: *Juridical Blockchain Recorder* mit Echtzeitanforderungen Dieser Use Case beschreibt im Kern die Anforderungen für das *RailChain*-Projekt, wie die DLT erforscht werden soll und mit welcher Strategie die bestehenden Anforderungen zumindest vergleichsweise erfüllt werden können.

3.1 Systemüberblick

Das folgende Kapitel gibt einen Überblick über das System Zug auf einer stark simplifizierenden, aber für die weiteren Betrachtungen ausreichenden Basis, um den Projektansatz und die Anforderungen verstehen zu können. Die folgenden Abbildungen stellen eine einheitliche Grundlage für den Use Case 1 dar und umfassen die wesentlichen Komponenten im Zusammenhang mit der properAsset Identity.

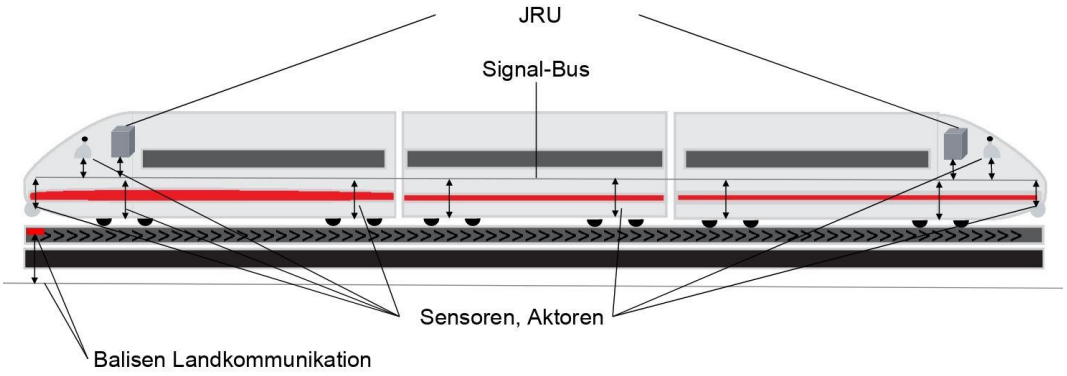

Abbildung 3.1: Schematischer Überblick: Sensoren, Aktoren und Datenbusse im Zug

Ein Zug besteht heutzutage aus vielen wichtigen Steuerkomponenten, Sensoren und Aktoren, die untereinander durch einen Signal-Bus vernetzt sind. Die abgebildeten JRUs sind in der Regel bei Triebzügen des Personenverkehrs doppelt vorhanden und arbeiten aus Gründen der Redundanz parallel.

Auf dem Signal-Bus heutiger Züge sind bei weitem nicht alle digital verfügbaren Informationen des Zuges vorhanden. Die Historie des Zugbusses beginnt bei der Vernetzung sicherheitskritischer Komponenten, sodass der Signal-Bus auch heute noch kritischer Teil der Sicherheitsarchitektur ist. Die verfügbaren Informationen sind ein Kompromiss aus Relevanz der Information und Kapazität der Bus-Technik. Eine weitere Rolle spielen dabei die auf sehr lange Nutzungsdauer von Zügen und die aus dem Sicherheitsbedürfnis öffentlicher Verkehrsmittel begründete sehr strenge Regulierung. Diese Faktoren und der Anspruch auf Rückwärtskompatibilität führen zu einem anderen Innovationstempo bei der Verwendung neuer Signal-Bus-Techniken im Vergleich zu anderen Industriezweigen.

Das folgende Bild veranschaulicht, dass es verschiedene Topologien der Vernetzung innerhalb eines Zuges gibt:

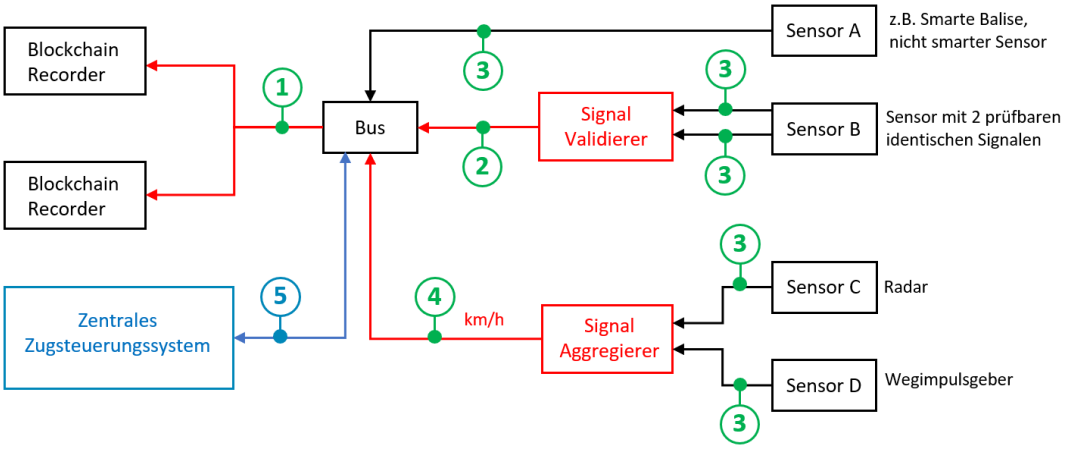

Abbildung 3.2: Schematische Darstellung zur Verortung der Sensoren

1. Daten zwischen dem *Blockchain Recorder* (BR) und Bus: nur aggregierte Daten vorhanden
2. Daten zwischen Bus und validiertem Signal: validierte Rohdaten vorhanden
3. Assets zwischen Sensor und Bus: reine Sensor-Rohdaten vorhanden
4. Assets zwischen Bus und aggregierten Daten: teilaggregierte Daten vorhanden
5. Assets zwischen Bus und zentraler Zugsteuerung

Für eine auf DLTs basierende Datenaufzeichnung sollten mindestens 4, besser 5 oder mehr Einheiten auf dem Zug verteilt sein. Diese Einheiten verfügen alle über Anschluss an den Zugbus und damit gleichberechtigten Zugang zu den Informationen auf dem Bus. Alle am Zugbus angeschlossenen Datenquellen können damit für eine Aufzeichnung herangezogen werden.

Die DLT bietet über sog. Konsensmechanismen die Funktion, einen geprüft abgeglichenen Informationsstand auf den Knoten zu gewährleisten bzw. Abweichungen von diesem zu erkennen. Im Rahmen des Projektes soll geklärt werden, wie der Konsens zwischen allen beteiligten Knoten sichergestellt werden kann. Diese Anforderungen sind neuartig und daher bei den bisherigen Architekturen von Fahrzeug-IT nicht berücksichtigt.

3.1.1 Zugbus

Das *Train Communication Network* (TCN) ist eine herstellerunabhängige, nach IEC 61375-1 [32] genormte Netzwerktopologie. TCN setzt sich aus dem Fahrzeugbus und dem überlagerten Zugbus zusammen. Die genaue technische Ausführung wird durch die Norm nicht festgelegt. Als Fahrzeugbus kommt der *Multifunction Vehicle Bus* (MVB), das *Ethernet Consist Network* (ECN) oder *CANopen* zum Einsatz, wobei innerhalb eines Zuges auch eine Mischung verschiedener Fahrzeugbusse möglich ist. Er verbindet die Subsysteme innerhalb einer kuppelbaren Zugeinheit, wie z. B. innerhalb eines Einzelwagens. Der Zugbus ist als *Wire Train Bus* (WTB) oder *Ethernet Train Backbone* (ETB) ausgeführt. Er verbindet die einzelnen Wagen bzw. Teilzüge miteinander über Gateways. Zur Erhöhung der Verfügbarkeit sind beide Bussysteme redundant ausgeführt. Zusätzlich zu den beiden nach TCN spezifizierten Bussen können Subsysteme herstellerspezifische Bussysteme innerhalb ihrer Grenzen nutzen. Je nach Notwendigkeit bzw. Projektierung wird dieses Bussystem über ein Gateway mit dem Fahrzeugbus verbunden. In Abbildung 3.3 ist eine solche Bustopologie beispielhaft dargestellt. Neben den gezeigten Subsystemen können noch weitere vorhanden sein.

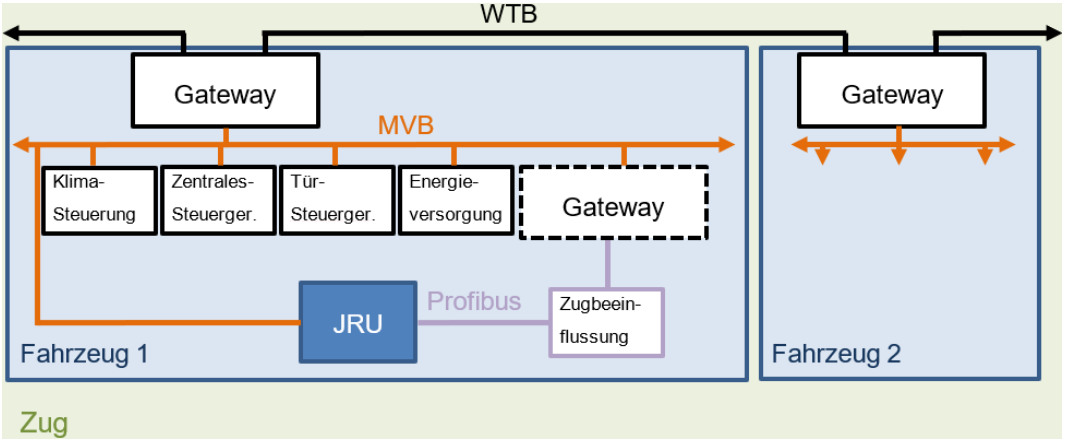

Abbildung 3.3: Beispiel einer Bustopologie, eigene Darstellung nach [35]

Da der MVB für den *Juridical Recorder* eine besondere Rolle einnimmt, wird er in diesem Abschnitt genauer beschrieben. Auf dem MVB werden drei verschiedene Datenarten übertragen:

- **Prozessdaten**

  Prozessdaten sind kurze, meist zeitkritische Daten von max. 32 Byte Länge, die periodisch, verbindungs- und quittierungslos in festen, projektierbaren Abständen als Datagramdienst übertragen werden.

- **Messagedaten**

  Messagedaten sind beliebig lange, weniger zeitkritische Daten, die nur auf Aufforderung übertragen werden, wie etwa spezielle Diagnosedaten. Messagedaten werden verbindungsorientiert übertragen und sind nicht echtzeitfähig. Aufgrund ihrer Länge werden sie oft in mehreren Telegrammen von 32 Byte übertragen. Um festzustellen, ob z. B. Diagnosedaten vorliegen, startet der Busmaster Ereignisrunden, in welchen er dies abfragt und Geräte auswählt, die zuerst Messagedaten senden.

- **Kontrolldaten**

  Kontrolldaten sind kurze Telegramme zur Steuerung der Busfunktionalität. Über sie wird der Status der Busteilnehmer abgefragt. Außerdem dienen sie zur Weitergabe der Busadministrator-Funktionalität. Sie besitzen eine Länge von 16 Bit.

Die Datenübertragung erfolgt in Basisperioden, in welchen der Busadministrator zyklisch Daten-Anfragen (sog. polling) an die Daten-Quellen (sog. Slaves) versendet. Eine Basisperiode teilt sich in verschiedene Phasen auf. Prozessdaten werden in der periodischen, Message- und Kontrolldaten in der sporadischen Phase gesendet. Um einen zeitgleichen Beginn der darauffolgenden Basisperiode zu gewährleisten, befindet sich an deren Ende eine Sicherheitszeit. Abbildung 3.4 stellt eine Basisperiode exemplarisch dar. Die sporadische Zeit wird hierfür auf den nach IEC 61375-3-1 [34] genannten Standardwert von 350 µs gesetzt. Zu beachten ist, dass die sporadische Zeit auch dynamisch angepasst werden kann, um z. B. eine größere/-geringere Anzahl von Prozess-/Messagedaten zu übermitteln. In der Darstellung ist eine Basisperiode des MVB dargestellt, bestehend aus Telegrammen für Prozessdaten, Kontrolldaten und Messagedaten. Die Längen der Basisperiode sowie der periodischen und sporadischen Phase sind exemplarisch anhand der in IEC 61375-3-1 [34] genannten Standardwerte gewählt.

Für die Datenübermittlung wird die zur Verfügung stehende Zeit nun mit Telegrammen gefüllt. Jedes Telegramm besteht dabei zunächst aus einem Masterframe von 40 Bit Größe. Dieses wird vom Busmaster versendet und beinhaltet die Anfrage an die einzelnen Busteilnehmer. Daran angehängt ist das sog. *Slaveframe*, welches je nach Telegrammtyp eine unterschiedliche Größe aufweist und die Übertragung von Nutzdaten zulässt.

Für ein Prozessdatentelegramm können Nutzdaten der Längen 16, 32, 64, 128 oder 256 Bit übertragen werden. Durch weitere Telegrammbestandteile, wie Prüfsummen und Start/Endanzeiger, kann das Slaveframe eine Größe zwischen 35

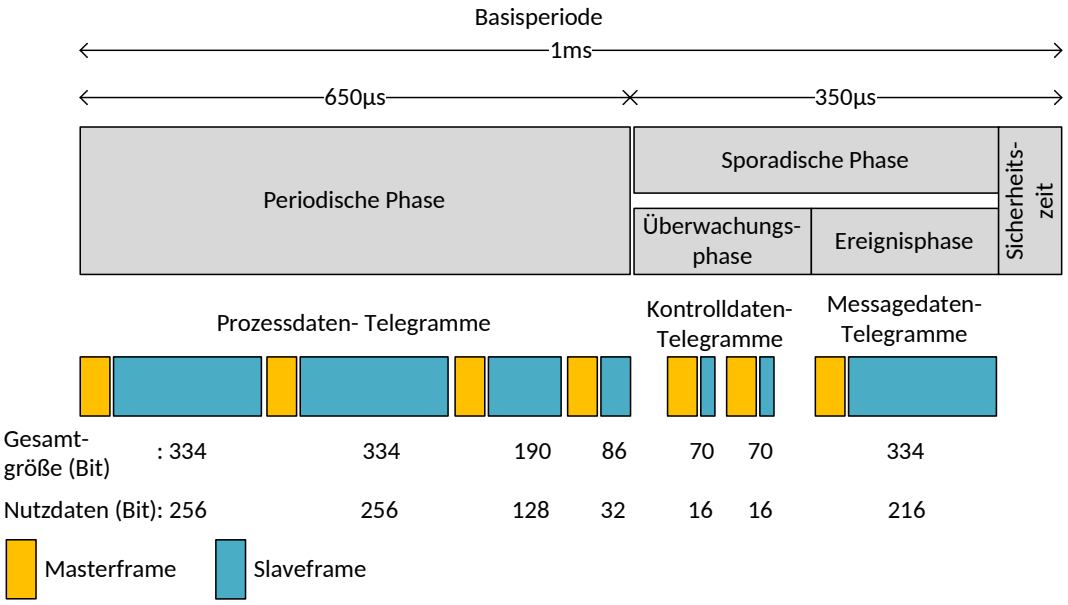

Abbildung 3.4: Darstellung einer MVB-Basisperiode und die Aufteilung der Phasen

Bit und 299 Bit annehmen, abhängig von der gewählten Nutzdatengröße. Messagedaten haben eine konstante Größe von 299 Bit, wovon 216 Bit für Nutzdaten verwendet werden können. Bei einer geringeren Nutzdatengröße wird der übrige Bereich mit Nullen gefüllt. Kontrolldatentelegramme haben ebenfalls eine konstante Größe, bieten jedoch nur 16 Bit Nutzdaten, sodass sich die Slaveframegröße auf 35 Bit beläuft.

3.1.2 Blockchain und *Asset Identity* – Allgemeine Erklärung

Blockchain bzw. DLT ist eine digitale Technologie, die Kryptografie-, Datenmanagement-, Netzwerk- und Anreizmechanismen kombiniert, um die Überprüfung, Ausführung und Aufzeichnung von Transaktionen zwischen Parteien zu unterstützen. Ein Blockchain-Ledger ist eine Liste (die „Kette“, engl. „chain“) von Gruppen („Blöcke“) von Transaktionen. Parteien, die eine Transaktion vorschlagen, können sie einem Pool von Transaktionen hinzufügen, die in der Blockkette erfasst werden sollen. Verarbeitungsknoten innerhalb des Blockchain-Systems nehmen einige dieser Transaktionen entgegen, überprüfen ihre Integrität und zeichnen sie in neuen Blöcken in der Blockkette auf. Der Inhalt des Blockchain-Ledgers wird auf mehrere räumlich verteilte Verarbeitungsknoten repliziert. Diese Verarbeitungsknoten betreiben gemeinsam das Blockchain-System, ohne die zentrale Kontrolle eines einzelnen vertrauenswürdigen Dritten. Das Blockchain-System stellt jedoch sicher, dass alle Knoten letztendlich einen Konsens über die Integrität und den gemeinsamen Inhalt des Blockchain-Ledgers erzielen [54].

Der oben beschriebene Technologiemix impliziert die folgenden nützlichen Eigenschaften:

1. Jeder beteiligte Knoten verfügt nach dem durchgeführten Konsens über den kompletten Datenbestand mit dem Wissen, dass dieser auf den anderen Knoten identisch ist.

2. Durch die Blockkette ist der technische Aufwand die einmal erfassten Daten nachträglich zu manipulieren hoch. Es müssten alle den manipulierten Daten folgende Daten auf allen beteiligten Knoten gleichzeitig und kryptografisch valide manipuliert werden. Der dadurch auch wirtschaftlich hohe Aufwand verhindert Manipulationen.

3. Die Herkunft der Transaktionen (Daten), die in das System eingebracht werden, ist an eine kryptografisch eindeutige Identität der Geräte oder Assets geknüpft. Sofern die privaten Schlüssel unzugänglich durch Dritte gehalten werden, ist die Quelle der Daten eindeutig nachvollziehbar.

Die Eigenschaften 1 und 2 bilden die Grundlage für die Idee, DLT als Ersatz für die gerichtsfeste Aufzeichnung von Fahrdaten von Zügen in Betracht zu ziehen. Die Verteilung der Informationen erfüllt die hohen Anforderungen an die Verfügbarkeit und die Integrität erfüllt die Anforderungen an die Unveränderlichkeit, an denen heutige JRUs jeweils gemessen werden.

Die 3. Eigenschaft eröffnet neue Möglichkeiten im Bahnbetrieb in Bezug auf die Nachvollziehbarkeit und Zuordnung von Daten, die im Rahmen der fortschreitenden Digitalisierung ohnehin anfallen und anfallen werden. Die Einführung einer eindeutigen digitalen Identität einer jeden Informationsquelle lässt Rückschlüsse auf einzelne Bauteile, deren Kreisläufe und deren Betriebszustand zu, die durch eine deutlich verbesserte Datenlage einen erheblichen Qualitätssprung bewirken können. Neben den passiven Eigenschaften der Nachvollziehbarkeit ist durch eine eindeutige *Asset Identity* auch die Möglichkeit gegeben, nicht nur Daten zu senden, sondern aktiv Transaktionen auszulösen. Durch die Eigenschaften von *Smart Contracts* können Automatismen geschaffen werden, die im Rahmen vorher gefasster Bedingungen ablaufen und immer nachvollziehbar sein werden.

Als Asset wird das physische Gerät, z. B. Türsteuergerät, Zugsteuerung, usw., welches digital abgebilidet werden soll, bezeichnet. Die *Asset Identity* ist die digitale Repräsentanz des physischen Gerätes, sie setzt sich aus zwei Komponenten zusammen:

1. Dem eindeutigen Identifizieren (dezentrale Identität)

2. Verifizierbare Daten (Nachweisbare Zertifikate)

    a. Statische verifizierbare Daten (z. B. Hersteller, Eigenschaften des Gerätes)

    b. Dynamische verifizierbare Daten (z. B. Aufzeichnungsdaten von Sensoren)

3.1.3 Digitales Testfeld

Das Forschungsvorhaben prüft die Anwendung und das Verhalten der angewandten Technologie zunächst unter den Laborbedingungen der Konsortialpartner des Hasso-Plattner-Instituts (HPIs) und der TU Braunschweig (TU BS).

Hier steht im Vordergrund, schnell und flexibel Hardware- und Softwarekomponenten auszuprobieren, aufeinander abzustimmen und deren Verhalten unter Stresssituationen zu erfahren. Dabei steht im Vordergrund, das Latenzverhalten der eingesetzten Konsensmechanismen in den folgenden Grenzsituationen zu erforschen:

- Lastgrenzen auf der Dateneingangsseite
- Latenzen in den Netzwerkverbindungen der Knoten untereinander
- Störungseinwirkungen bzw. Ausfall der Netzwerkverbindungen oder ganzer Knoten

Dafür steht im HPI IoT-Labor (dem Labor für das *Internet of Things*) eine Testinfrastruktur zur Verfügung, mit der diese Szenarien unter Zuhilfenahme von Netzwerksimulatoren und Techniken der Fehlerinjektion reproduzierbar getestet werden können. Weiterhin gibt es einen von Siemens bereitgestellten Testaufbau eines MVB-Bussystems im Labor der TU Braunschweig, der Tests auf tatsächlich im Zug verbauten Rechnern ermöglicht und damit eine hohe Repräsentativität der Testergebnisse ermöglicht.

Neben den Verhaltenserfahrungen aus den Laborbedingungen spielt die praktische Anwendbarkeit in der Beurteilung der Eignung der DLT eine entscheidende Rolle. Daher hat das Projekt sich zum Ziel gesetzt ein möglichst reales Versuchsumfeld zu nutzen, um die praktische Tauglichkeit der erprobten Systemkomponenten unter Beweis zu stellen.

3.1.4 Versuchsträger *advanced TrainLab*

Das Projekt nutzt das Angebot der DB AG die Komponenten auf dem Versuchsträger *advanced TrainLab* verbauen zu können und so direkt unter den realen Bedingungen eines fahrenden Laborzuges das Verhalten und die Tauglichkeit zu testen [35].

Das *advanced TrainLab* besteht aus zwei Zügen der Baureihe VT605 (Ordnungsnummer 517 und 519) mit dieselelektrischem Antrieb und einem WTB bzw. MVB, der die Datenquelle für die zu speichernden Betriebsdaten darstellt.

Die Bordarchitektur und die konkrete Einbausituation sind durch das Projekt zu prüfen. Eine rückwirkungsfreie Nutzung ist zu gewährleisten.

Im Bereich ETCS/ERTMS existiert eine Spezifikation im Subset-027 [17] zum *Juridical Recorder* im Sinne der Unfalldatenspeicherung. Diese wird von den Herstellern durch eine JRU bereitgestellt, die neben den Ereignissen des ETCS-Hauptrechners (*European Vital Computer*, EVC) auch die Signale der Class-B-Systeme speichert.

Abbildung 3.5: Das *advanced TrainLab* der DB

Derzeit verwendete JRUs sind an verschiedene Signalquellen angeschlossen und schreiben relevante Signale und Nachrichten in ihren internen Speicher. Die meisten Informationen werden dabei über das TCN bezogen, jedoch kann die JRU je nach Projektierung auch weitere Schnittstellen zu anderen Teilsystemen und sogar analoge Eingänge besitzen.

Die Fälschungssicherheit des Datenspeichers steht hierbei im Vordergrund. Neben allen externen Ereignissen wie erkannten Eurobalisen und internen Ereignissen, wie z. B. betätigte Bremsen, wird mindestens alle fünf Sekunden eine allgemeine Zustandsinformation gespeichert (mit Zeitstempel, Zugposition, Geschwindigkeit, Betriebssystemversion, ETCS-Level und -Modus). Eine wichtige Anforderung an die JRU sind die Echtzeitanforderungen (< 1 Sekunde Block-Zyklus-Zeit). Diese Anforderung muss in jedem Falle auch vom *Juridical Blockchain Recorder* (JBR) erfüllt werden.

In Abbildung 3.6 ist am Beispiel einer heute verbauten JRU erkennbar, welche weiteren Vorteile ein JBR bringen könnte. Derzeit ist eine Datenabgriff nur über eine RS232-Schnittstelle spezifiziert [35]. Mit einem JBR könnte ein vereinfachter Zugriff auf die Daten ermöglicht werden sowie eine höhere Vertrauenswürdigkeit der Daten, da die Daten in einem JBR unveränderlich und kryptographisch gesichert abgelegt werden. Dies könnte auch den Wert der Daten steigern sowie den Einstieg in eine wirtschaftliche Verwertung der Daten ebnen.

Nach Kenntnis der Autoren ist dies der erste Anwendungsfall für DLTs für Echtzeitanwendungen im Eisenbahnbetrieb (Operational Technology).

3.1.5 Vorteile des *Juridical Blockchain Recorder*

Das Ziel dieses Use Cases bestand darin die Funktionen der JRU durch eine blockchainbasierte Software abzubilden. Durch Einsatz dieser Software auf im Zug vorhandenen Rechnern mit freien Rechenkapazitäten kann die physische JRU eingespart werden. Dadurch sollen sich folgende Vorteile ergeben:

1. Dadurch, dass mehrere im Zug verteilte Rechner die Aufgabe der Datenaufzeichnung als verteiltes System übernehmen, steigt die Resilienz der Aufzeichnungsdaten gegen Verlust durch Unfälle. Die Wahrscheinlichkeit, dass

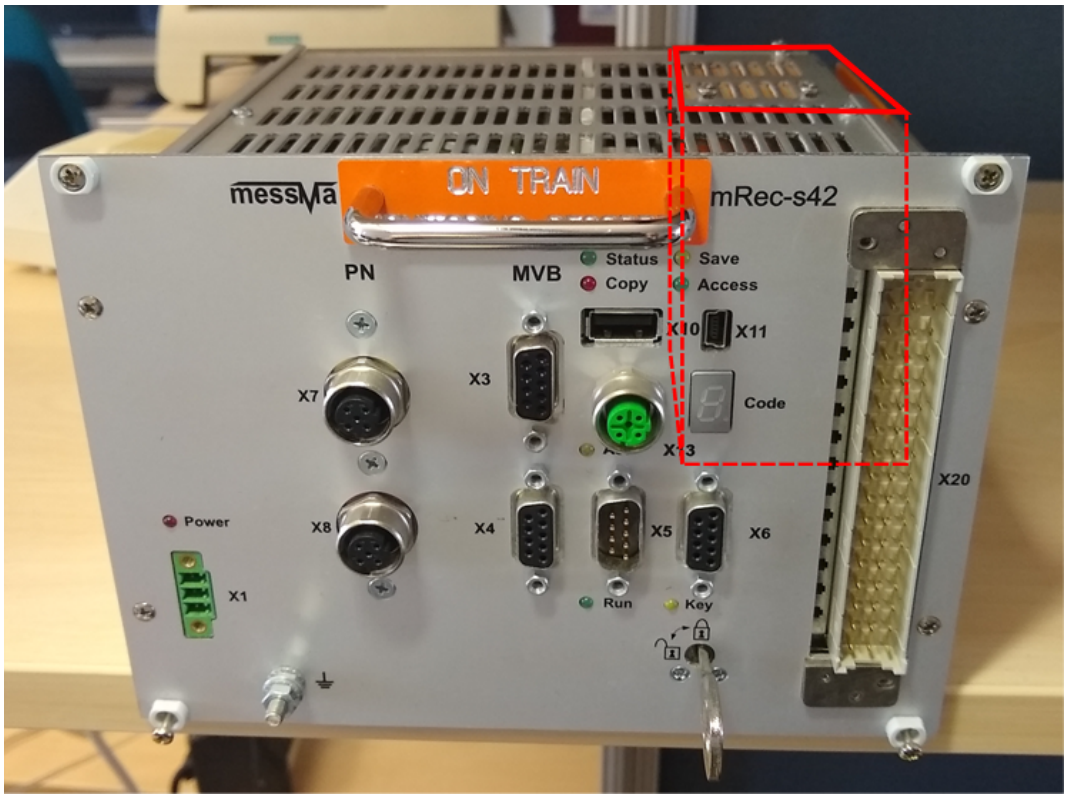

Abbildung 3.6: JRU mit USB- und seriellen Schnittstellen zum Auslesen von Daten

sich zumindest ein Teil der Rechner in einem ausreichend unversehrten Teil des Zuges befindet, ist sehr hoch.

2. Gleichzeitig steigt im verteilten System die Fehlertoleranz, weil sich die aufzeichnenden Rechner gegenseitig überprüfen. Entsprechende Einigungsprotokolle greifen im Fehlerfall und sichern die Konsistenz der gespeicherten Daten ab.

3. Die Speicherung in Form einer Blockchain vereinfacht das drahtlose Hochladen der auf dem Zug befindlichen Fahrdaten auf die zur Auswertung und Archivierung genutzten Server. Die verschlüsselten Daten können über unsichere Netzwerke wie das Internet verteilt werden. Dadurch wird die Zugänglichkeit der aufgezeichneten Fahrdaten verbessert.

4. Durch die intrinsische Blockchain-Verschlüsselung steigt die Sicherheit der gespeicherten Fahrdaten gegenüber Manipulation. Dies betrifft sowohl die auf dem Zug gespeicherten als auch die auf stationären Servern archivierten Daten.

Da die Aufzeichnung der Fahrdaten von mehreren bereits existierenden Rechnern als sekundäre Funktion übernommen wird, kann der Einsatz einer speziellen

Recheneinheit mit gehärtetem Speicher eingespart werden. Dadurch steigt die Wirtschaftlichkeit des *Juridical Recorder*s.

3.2 Use Case 1: *Asset Identity* inkl. *Asset Tracking*

Das System Bahn hat aus Gründen der Akzeptanz in der Gesellschaft sehr hohe Anforderungen an die Sicherheit des Gesamtsystems. Diese hohe Verantwortung wird durch staatlich unterstellte Überwachungsorgane wahrgenommen, die Vorgaben für die Standardisierung und Zulassung aller am Schienenverkehr teilnehmenden Systemkomponenten machen.

Der Zug, wesentlicher Teil dieses Systems, besteht selbst aus vielen technischen Komponenten. Der ersten Betriebszulassung gehen aufwändige Modell- und Einzelprüfungen voraus und für den Erhalt der Zulassung sind Vorgaben für die Wartung einzuhalten.

Die korrekte Zusammensetzung der Zugkomponenten und der Nachweis darüber sind zurecht wesentlicher Bestandteil und Aufwandstreiber aller für die Planung und den Betrieb erforderlichen Prozesse. Das recht komplexe Zusammenspiel aus Aufsichtsbehörden, Herstellern und Betreibern von rollender (Eisenbahnverkehrsunternehmen, EVU) und stationärer (Eisenbahninfrastrukturunternehmen EIU) Infrastruktur führt zu einem sehr stark regulierten Umfeld.

Die DLT und das recht neue Konzept der *Decentralized Identifiers* [27] (DIDs) und *Verifiable Credentials* [5] (VCs) haben das Potenzial, den Anforderungen an Sicherheit und Nachweispflicht nachzukommen und zusätzlich eine bisher nicht darstellbare Flexibilität und Effizienz zu ermöglichen. Deshalb sollten die Standards DIDs und VCs des World Wide Web Consortium (W3C) für eine Eignung bei der Umsetzung in Betracht gezogen werden. Der DID ist eine Art von Kennung für Assets, der eine überprüfbare, dezentrale digitale Identität ermöglicht. VCs sind verifizierbare Eigenschaften, welche als Datensatz eine digitale Identität beschreiben. Diese Eigenschaften können Herkunftszertifikate, Berechtigungen oder Lebenszyklusdaten sein, die mit einer *Asset Identity* verknüpft sind. Das Projekt *RailChain* möchte diese Technologie mit konkreten Anwendungsbeispielen ausprobieren und die Realisierbarkeit überprüfen.

Deshalb ist es das Ziel des Use Case 1, ein *Asset-Identity*-Konzept auf Basis von DIDs und VCs zu definieren, welches die Anforderungen der anderen Use Cases erfüllt. Es soll zusätzlich ein kompletter Lebenszyklus der einzelnen Assets aufgestellt werden können. Diese Betrachtungen werden nur im Labor oder in vom bestehenden Zugsystem entkoppelten Komponenten abbildbar sein. Die angestrebten Funktionen greifen so weit in den Sicherheitsbereich bestehender Schienenfahrzeuge ein, dass eine Integration dieser neuen Technologie ein Erlöschen der Betriebserlaubnis zur Folge hätte.

Der untersuchte Sachverhalt berührt die Domäne der Systeme für *Enterprise Asset Management*, die in sehr unterschiedlichen Detailausprägungen und Zwecken existieren (unter anderem Infrastrukur- und Software-*Asset*-Management). Sicher-

lich hat das Konzept der DIDs und VCs das Potenzial, auf alle o. g. Systeme Auswirkungen zu haben.

3.2.1 Funktionale Anforderungen

Die nachfolgend dokumentierten funktionalen Anforderungen dienen dazu, einen Rahmen für die technische Evaluation von DLT im Zusammenspiel mit DIDs und VCs zu setzen. Das Ergebnis sollte die Machbarkeit und Eignung im Fokus behalten, sowie ggf. auch Alternativen aufzeigen und Grenzen der Umsetzbarkeit dokumentieren.

Die benötigten Funktionen werden in Use Cases beschrieben, die sich am Lebenszyklus eines Assets orientiert und an der Anforderung Asset-Informationen für Folgeprozesse zu speichern und verfügbar zu machen.

Die wesentlichen Nutzenversprechen, die anhand des Projektes überprüft werden sollen, sind:

- Ableitung einer validierten Asset-Historie
- Vom Asset übermittelte Daten sind eindeutig zuordnungsfähig und fälschungssicher
- Zugzustand ist validiert und nachvollziehbar
- Automatisierbare Wartungsvorgänge (Ausbau, Einbau, Wartung)
- Zugriff auf gespeicherte Asset-Informationen ist durch gesicherte Autorisierung geregelt

Die validierte Asset-Historie ist ein wertvoller Bestandteil der Betrachtungen, weil der Austausch von Lebenszyklus und Detaildaten über das Asset im Interesse der EVUs und der Hersteller der Assets ist. Produktverbesserungen lassen sich besser umsetzen, wenn die Schwachstellenanalyse auf eine bessere und vertrauensvolle Datenbasis aufbauen kann.

Da das Forschungsprojekt *RailChain* sich gemäß dem Antrag mit drei verschiedenen **Use Cases**:

1. *Asset Identity*
2. *Blockchain Recorder*
3. *Juridical Blockchain Recorder*

beschäftigt, sprechen wir im Folgenden von verschiedenen Szenarien innerhalb von Use Case 1 – *Asset Identity*, die wir anhand von Use-Case-Diagrammen erklären.

Bei den Use-Case-Diagrammen in Abbildung 3.7, Abbildung 3.8 und Abbildung 3.9 liegt das Hauptaugenmerk in der Funktionalität des Systems aus Sicht des Nutzers. Der Fokus der Szenarien liegt auf dem Lebenszyklus der Assets.

Der *Asset Life Cycle* (dt. Asset-Lebenszyklus) wird in drei Hauptszenarien unterteilt: Vor dem Betrieb, im Betrieb und nach dem Betrieb.

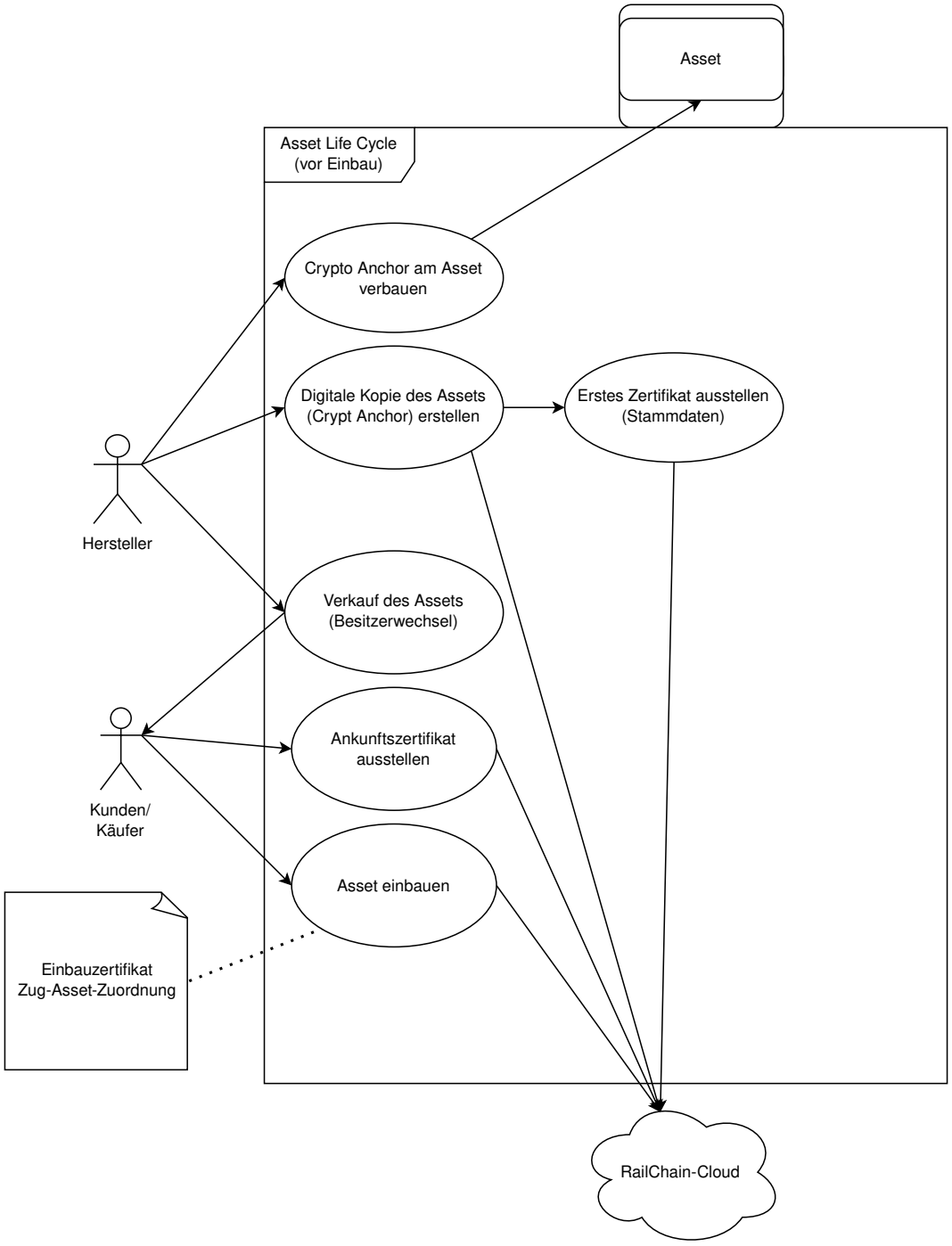

Abbildung 3.7: *Asset Life Cycle* I – optionale Szenarien sind nicht im Diagramm enthalten

3.2.1.1 Szenario *Asset Life Cycle* (vor Einbau) – Verpflichtend

Kurzbeschreibung	Dem Asset werden vor dem Einbau für jeden Schritt entlang desLebenszyklus, bis zum initialen Einbau, Zertifikate als VCs ausgestellt, um deren Historie aufzuzeichnen.
Akteure	Hersteller, Kunde
Vorbedingungen	Asset erstellen
Fachlicher Auslöser	Erstellung des Assets
Ergebnis	Die digitale Kopie des Assets (DID) hat folgende Zertifikate ausgestellt bekommen, bevor es in dem Zug verbaut wird: - Geburtszertifikat (Master Data) - Shipments Zertifikat (Outbound) - Ankunftszertifikat (Inbound) - Einbau Zertifikat (Zugbindung)

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0001:01`
Titel	**Asset – Digitale Kopie des Assets erstellen**
Beschreibung	Das Asset bekommt eine eindeutige Kennung (DID) und ein Herstellerzertifikat als VC, das seine Eigenschaften und Betriebsanforderungen definiert. Das Asset existiert real und erfüllt die nachfolgenden Anforderungen.
Hinweise	—
Bewertung	Erfüllt

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0001:01`
Titel	**Asset – Crypt Anchor verbauen**
Beschreibung	Das Asset bekommt ein kryptographiefähigen Hardware-Chip verbaut auf dem ein Schlüsselpaar (öffentlicher und privater Schlüssel) erzeugt wird. Das Asset hat einen privaten Schlüssel, der zur Signatur oder Entschlüsselung herangezogen werden kann. Der private Schlüssel sollte nur im Asset vorhanden sein und gegen Kopieren und Auslesen geschützt sein. Es ist zu evaluieren, ob ein Key-Rotation-Mechanismus anzuwenden ist, um die Kompromittierbarkeit der Signatur zu erschweren. Der private Schlüssel kann bei hohen Sicherheitsanforderungen auch über ein HSM (Hardware Secure Module) verwaltet werden, für den Demonstrator im Projekt ist das keine zwingende Voraussetzung.
Bewertung	Nicht erfüllt.
Hinweise	Das ist eine Anforderung, die im Rahmen der Demonstrierbarkeit des Funktionsprinzips nicht umgesetzt werden muss.

Identifikator	`#REQ-RC_UC1_AssetsLebenszyklus_#DEF0004:01`
Titel	**Asset – Erstes Zertifikat ausstellen (Stammdaten)**
Beschreibung	Erstellen eines Zertifikates, das vom Hersteller als VC ausgestellt wird und die Eigenschaften des Assets spezifiziert und dessen Betriebsanforderungen beschreibt. Die Signatur des Herstellers im VC ist von Dritten überprüfbar und bezeugt die Richtigkeit der Informationen.
Hinweise	—
Bewertung	Erfüllt Siehe Herstellerzertifikat

Identifikator	`#REQ-RC_UC1_AssetsLebenszyklus_#DEF0004:01`
Titel	**Asset – Verkauf des Assets (Besitzerwechsel)**
Beschreibung	Der Wechsel des Besitzers des Assets ist im digitalen Abbild hinterlegt und ändert dadurch den Kontrolleur des digitalen Abbilds. Der Hersteller überträgt die Rechte des digitalen Abbildes auf den Eigner, der Eigner ändert den besitzanzeigenden öffentlichen Schlüssel im digitalen Abbild. Ggf. ist ein Key-Rotation-Vorgang für den privaten Schlüssel des Assets zu prüfen. Besitzerwechsel ist vollzogen; Hersteller hat keine Kontrolle mehr über Asset und digitales Abbild.
Hinweise	Es ist durch einen öffentlichen Schlüssel auf dem digitalen Abbild vorgegeben, dass der Inhaber des zugehörigen privaten Schlüssels Schreibrechte auf das digitale Abbild hat.
Bewertung	Erfüllt Siehe Prozess Besitzerwechsel

Identifikator	`#REQ-RC_UC1_AssetsLebenszyklus_#DEF0004:01`
Titel	**Asset – Ankunftszertifikat ausstellen**
Beschreibung	Erstellen eines Zertifikates, das vom Käufer als VC ausgestellt wird und die Ankunftsdaten des Assets spezifiziert und dessen Zustand beschreibt. Die Signatur des Käufers im VC ist von Dritten überprüfbar und bezeugt die Richtigkeit der Informationen.
Hinweise	Erst nachdem das Zertifikat ausgestellt wurde wird der Abrechnungsprozess in die Wege geleitet.
Bewertung	Erfüllt Siehe Prozess Besitzerwechsel

Identifikator	`#REQ-RC_UC1_AssetsLebenszyklus_#DEF0004:01`
Titel	**Asset – Asset einbauen**
Beschreibung	Erstellen eines Zertifikates, das vom Service-Mitarbeiter als VC ausgestellt wird und den Installations Zeitpunkt des Assets spezifiziert und dessen Zustand beschreibt. Die Signatur des Service-Mitarbeiter im VC ist von Dritten überprüfbar und bezeugt die Richtigkeit der Informationen. Außerdem wird durch das Asset eine Installationsnachricht als VC ausgestellt.
Hinweise	—

Bewertung	Erfüllt Siehe Prozess Wartung

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0007:01`
Titel	**Zugsteuerung einbauen und initialisieren – optional**
Beschreibung	Die Zugsteuerung ist ein besonderes Asset, da es wesentliche Kontrollfunktionen bei der Verwaltung der im Zug vorhandenen Assets ausübt. Das betrifft sowohl die verifizierbare Zulassung der Assets im System als auch der Überwachung des Betriebszustandes.
Hinweise	Die Zugsteuerung wird den gültigen physikalischen und elektrischen Anforderungen eines Zugsystems nach in einer Zug-/Service-Station eingebaut, mit der Energieversorgung und mit dem Zugbus verbunden. Ein manueller Auslöser startet die Zugsteuerung. Das Asset prüft sein Betriebsumfeld und bezieht die Verifikationsdaten zum Abgleich des eigenen Herstellerzertifikates aus dem Netz. Die Zugsteuerung prüft die Informationen und initialisiert und speichert das Herstellerzertifikat im Zwischenspeicher für Zertifikate.
Bewertung	Nicht erfüllt

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0007:01`
Titel	**Auslöser für den Start der Zugsteuerung – optional**
Beschreibung	Die Zugsteuerung hat einen Auslöser, den der Service-Mitarbeiter betätigen kann, nachdem er das Asset mit dem Zug ordnungsgemäß (Stromversorgung, Bus) verbunden hat.
Hinweise	—
Bewertung	Nicht erfüllt

3.2.1.2 Szenario Asset einbauen – verpflichtend

Kurzbeschreibung	Das Asset wird in ein Umfeld eingebaut und registriert, in dem es funktionieren soll.
Akteure	Service-Mitarbeiter
Vorbedingungen	Es gibt ein Asset „Zugsteuerung“, das die Registrierung durchführt.
Fachlicher Auslöser	Einbau eines Assets in ein dafür vorgesehenes Betriebsumfeld (z. B. Zug- oder Service-Station)

Hauptszenario	Der Servicemitarbeiter liest den öffentlichen Schlüssel des Assets und initialisiert den Registrierungsprozess.
Ergebnis	Das Asset ist eingebaut und hat ein Einbauzertifikat erhalten.

3.2.1.3 Szenario Asset einbauen – optional

Kurzbeschreibung	Das Asset wird in ein Umfeld eingebaut und registriert, in dem es funktionieren soll.
Akteure	Service-Mitarbeiter
Vorbedingungen	Es gibt ein Asset „Zugsteuerung“, das die Registrierung durchführt.
Fachlicher Auslöser	Einbau eines Assets in ein dafür vorgesehenes Betriebsumfeld (z. B. Zug- oder Service-Station)
Hauptszenario	Das Asset wird den gültigen physikalischen und elektrischen Anforderungen eines Zugsystems nach in einer Zug-/Service-Station eingebaut, mit der Energieversorgung und mit dem Zugbus verbunden. Ein manueller Auslöser durch den Service-Mitarbeiter startet die Registrierung des Assets. Das Asset prüft sein Betriebsumfeld, weist sich durch das Herstellerzertifikat bei der Zugsteuerung aus und beantragt die Registrierung. Die Zugsteuerung prüft die folgenden Informationen: • Die nötigen Datensätze zur Prüfung des Herstellerzertifikates und öffentlich verfügbare Asset-Informationen werden vom Asset Repository angefordert. • Die Zugsteuerung überprüft das Herstellerzertifikat gegen den Bauplan des Zuges ob das Bauteil vorgesehen ist. Wenn beide Prüfungen erfolgreich abgeschlossen sind, wird das Asset im Asset-Register als vorhanden gespeichert und dem Asset die Betriebsfreigabe erteilt.
Ergebnis	Das Asset ist eingebaut, hat sein Umfeld und seine Funktion überprüft, ist registriert und hat seine Betriebsfreigabe erhalten.

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0010:01`
Titel	**Auslöser für den Registrierungsprozess**
Beschreibung	Das Asset hat einen Auslöser, der den Registrierungsprozess startet.
Hinweise	—
Bewertung	Nicht durchführbar wegen Wahrung der Rückwirkungsfreiheit. Eine Umsetzung in einem Demoumfeld ist leicht möglich.

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0011:01`
Titel	**Eigendiagnose Asset**
Beschreibung	Das Asset überprüft durch Eigendiagnose sein Betriebsumfeld (z. B. Versorgungsspannung, Temperatur, Sensoren, Kommunikation). Dieser Vorgang wiederholt sich ständig während des Betriebszustandes.
Hinweise	Gleiche Anforderung in Szenario Nutzung: Asset-Eigendiagnose durchführen
Bewertung	Nicht umgesetzt

3.2.1.4 Szenario Asset Warten – optional

Kurzbeschreibung	Das Asset wird durch einen Wartungsvorgang überprüft und ggf. verändert.
Akteure	Service-Mitarbeiter, Hersteller
Vorbedingungen	Das Asset muss in einem Umfeld eingebaut sein, das eine Kommunikation möglich ist. Das kann eine Service-Station sein, die speziell für die Wartung und Überarbeitung von Assets eingerichtet ist, oder ein eingebauter Zustand innerhalb eines Zuges sein.
Fachlicher Auslöser	Wartungsvorgang nach Wartungsplan oder nach Fehlfunktion
Hauptszenario	Während des Wartungsvorgangs von Assets werden folgende Arbeitsschritte durchgeführt: • Zertifikate werden überprüft • Software/Hardware wird aktualisiert • Neue Zertifikate werden ergänzt

Ergebnis	Das Asset ist entweder in einem ordnungsgemäßen Zustand mit gültigen Zertifikaten ausgestattet und wieder in den Lebenszyklus zurückgeführt oder es wird in den End-of-Life überführt.

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetLebenszyklus_#DEF0012:01`
Titel	**Wartungszertifikat ausstellen**
Beschreibung	Details zur Wartung müssen beschrieben werden: • Wann wurde die Wartung durchgeführt? • Welche Assets wurden verändert und wie? • Welche Softwareversionen befinden sich auf den Assets? • Wer hat die Wartung durchgeführt?
Hinweise	—
Bewertung	Erfüllt Siehe Prozess Wartung

Asset-Nutzung Die Nutzung des Assets ist von dem Konzept der DIDs entkoppelt. Die folgenden Anforderungen beschreiben die Funktionalität, die ein Asset benötigt, um z. B. Sensordaten unterschrieben an die BR zu senden. Außerdem wird der Prozess des Datenexports in die *RailChain*-Cloud analysiert.

3.2.1.5 Szenario *Asset Life Cycle* (im Betrieb) – Verpflichtend

Kurzbeschreibung	Die Assets signieren während des Betriebes alle Daten die von den Sensoren erfasst werden und senden sie an den BR.
Akteure	Zug, BR, Asset
Vorbedingungen	Asset erstellen
Fachlicher Auslöser	Inbetriebnahme
Ergebnis	Die Sensordaten befinden sich signiert auf dem BR und der *RailChain*-Cloud.

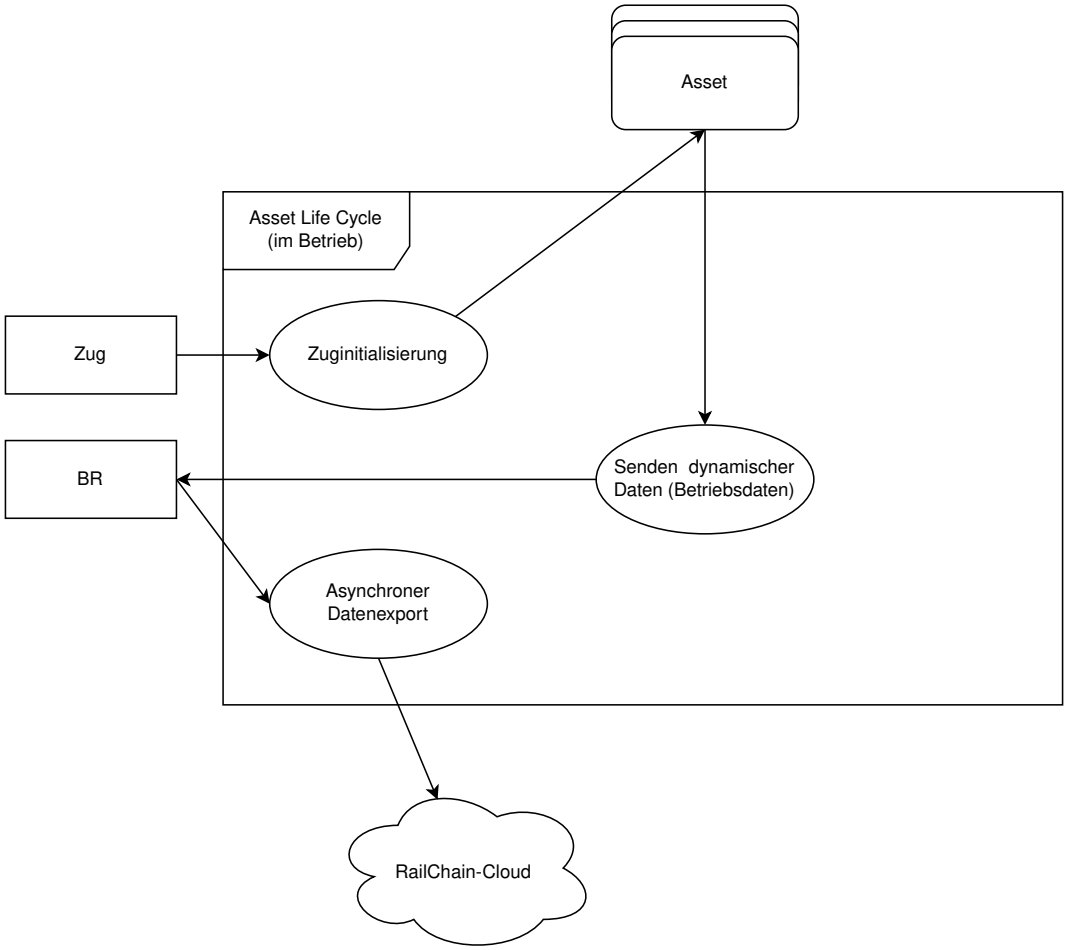

Abbildung 3.8: *Asset Life Cycle* II – optionale Anforderungen sind nicht im Diagramm enthalten

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0003:01`
Titel	**Senden dynamischer Daten (Betriebsdaten)**
Beschreibung	Die dynamisch ermittelten Sensordaten der Assets müssen mittels des privaten Schlüssels unterschrieben und an den BR gesendet werden.
Hinweise	Industrielle Standardverfahren können verwendet werden. Die Assets sind durch einen Schlüsselpaar identifizierbar und können die Sensordaten unterschreiben.
Bewertung	Nicht erfüllt. Wegen der Rückwirkungsfreiheit konnte diese Anforderung nicht umgesetzt werden. Zudem lassen die Standardfunktionen der DIDs keine Signaturen mit dem privaten Schlüssel zu.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0003:01`
Titel	**Zuginitialisierung**
Beschreibung	Das initiieren des Zuges aktiviert die Synchronisierung der einzelnen Assets. Das Senden der dynamischen Daten wird eingeleitet.
Hinweise	Nur beim Start des Zuges.
Bewertung	Nicht erfüllt. Wegen der Rückwirkungsfreiheit konnte diese Anforderung nicht umgesetzt werden.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0003:01`
Titel	**Asynchroner Daten Export**
Beschreibung	In einem regelmäßigen Abstand werden die auf dem Zug gesammelten Daten exportiert. Es muss sichergestellt werden, dass die Daten in der Cloud mit der Unterschrift der Assets verifizierbar abgelegt werden.
Hinweise	Der Datenexport ist notwendig um die Integrität des *Juridical Recorders* auf dem Zug zu gewährleisten.
Bewertung	Erfüllt im Rahmen des Use Case 2 und Use Case 3

3.2.1.6 Szenario Datenaufbereitung – verpflichtend

Kurzbeschreibung	Aufbereitung der Asset spezifischen Daten inkl. der Signaturen aus dem Datenstrom der BR-Daten (Zuordnung der Sensordaten zum Asset)
Akteure	Kunde (*RailChain*-Cloud)

Vorbedingungen	Die aufgezeichneten Daten der BR sind in der *RailChain*-Cloud verfügbar.
Fachlicher Auslöser	—
Hauptszenario	—
Ergebnis	—

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0001:01`
Titel	**Konsistenter Datenstand**
Beschreibung	Daten werden konsistent und kontinuierlich in der Cloud abgesichert.
Hinweise	—
Bewertung	Erfüllt im Rahmen des Use Case 2 und Use Case 3

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0001:01`
Titel	**Zuordnung der Daten**
Beschreibung	Die Daten können eindeutig dem digitalen Zwillingdes Assets zugeordnet werden.
Hinweise	Hierfür sollte der öffentliche Schlüssel bei der Transaktion übergeben werden.
Bewertung	Erfüllt im Rahmen des Use Case 2 und Use Case 3

3.2.1.7 Szenario Asset: Eigendiagnose durchführen – optional

Kurzbeschreibung	Jedes Asset benötigt ein individuelles Eigendiagnoseverfahren, um ausfallsicheren Betrieb zu gewährleisten. Die Prüfung der Sensormessung ist der Bestandteil der Eigendiagnose. Anhand eines Referenzwerts im Vergleich zu Messwert wird festgestellt, ob eine Störung des Assets vorliegt.
Akteure	Asset
Vorbedingungen	Eigendiagnoseverfahren muss dem Asset vorliegen
Fachlicher Auslöser	—
Hauptszenario	Durch die Eigendiagnose wird festgestellt, ob das Asset betriebsbereit ist.
Ergebnis	Das Protokoll der Eigendiagnose ist manipulationssicher und das Eigendiagnoseverfahren muss ausfallsicher verfügbar sein.

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0004:01`
Titel	**Eigendiagnoseverfahren**
Beschreibung	Das Eigendiagnoseverfahren wird vom Hersteller oder Eigentümer integriert. Die Referenzwerte für das Asset sind integrale Bestandteil.
Hinweise	Das Eigendiagnoseverfahren des Assets kann während Wartungsarbeiten integriert/modifiziert werden, oder initial bei der ersten Inbetriebnahme.
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0005:01`
Titel	**Eigendiagnoseverfahren Definition**
Beschreibung	Die Definition der Diagnose erfolgt in Abstimmung mit dem Hersteller bzw. der Betriebsführung des Eigentümers.
Hinweise	Die Definition kann angepasst werden, um bei technischer Umrüstung oder Erweiterungen am Asset alle Funktionen vollständig zu prüfen.
Bewertung	Nicht erfüllt, s. `#REQ-RC_UC1_AssetNutzung_#DEF0004:01`

3.2.1.8 Szenario Zugsteuerung: Fehlfunktion Asset behandeln – optional

Kurzbeschreibung	Nach Bewertung der Fehlfunktionen erfolgt eine priorisierte Sortierung der Fehlfunktionen. In dieser erfolgt eine automatische Bewertung des Störungsgrads mit anschließender Einleitung der Maßnahmen. Z. B. Bei sicherheitsrelevanten Vorfällen das Einleiten von Notbremsung/Langsamfahrt und Signalisierung an Triebfahrzeugführer (Tf).
Akteure	Zugsteuerung
Vorbedingungen	Festgelegte Maßnahmen die bei Störungen eingeleitet werden
Fachlicher Auslöser	Eintritt einer Fehlfunktion eines Assets
Hauptszenario	Automatische Sortierung der Fehlfunktion, die von Sensoren des Assets geliefert werden und Einleitung der Maßnahmen für die Störung
Ergebnis	Sicherer Zustand des Zuges ist bei jeder Art der Störung gewährleistet.

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0006:01`
Titel	**Vorgaben von Maßnahmen zur Behandlung der Störungen**
Beschreibung	Zu jeder auftretenden Störung im Zug, die von einem Sensor des Assets geliefert werden, muss eine geeignete Maßnahme zur Behandlung vorliegen.
Hinweise	—
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0007:01`
Titel	**Sortierfunktion der Störungen**
Beschreibung	Es muss eine automatische Sortierung nach Dringlichkeit anhand der Störungen erfolgen.
Hinweise	—
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

3.2.1.9 Szenario Asset: Daten signieren und senden – optional

Kurzbeschreibung	Die ermittelten Werte aus der Eigendiagnose werden mit dem privaten Schlüssel des Assets signiert und an den BR gesendet.
Akteure	Assets
Vorbedingungen	Privater Schlüssel des Assets muss vorhanden sein.
Fachlicher Auslöser	Eigendiagnoseverfahren
Hauptszenario	Nach Durchführung der Eigendiagnose dürfen ermittelte Werte nicht mehr verfälscht werden. Jede Diagnose muss mit einer separaten Signatur versehen werden und chronologisch nachvollziehbar sein.
Ergebnis	Manipulationssichere Übertragung der Daten von Assets

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0008:01`
Titel	**PKI-Verfahren**
Beschreibung	Für die Signatur der Daten ist der private Schlüssel eines Assets erforderlich.
Hinweise	—
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0009:01`
Titel	**Übertragung der Daten an BR**
Beschreibung	Manipulationssichere Übertragung der Daten an BR
Hinweise	—
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

3.2.1.10 Szenario Zuginitialisierung – optional

Kurzbeschreibung	Alle notwendigen technischen Funktionen wie Sensoren, Busleitung, Anbindungen an Schnittstellen für die Übertragung nach Außen (Infrastruktur DB Netz, Cloud, etc.) werden automatisch in den aktiven Zustand gebracht, ausgelöst durch den Tf.
Akteure	Tf
Vorbedingungen	Das System ist vollständig integriert und liegt betriebsbereit vor
Fachlicher Auslöser	Tf
Hauptszenario	Automatische Aktivierung des Systems, ausgelöst durch den Tf
Ergebnis	Störungsfreie und deterministische terminierende Aktivierung der Systeme. Feedback im Störungsfall

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0001:01`
Titel	**Authentifizierung des Tf**
Beschreibung	Initialisierungsprozess kann erst nach einer erfolgreichen Authentifizierung gestartet werden.

Hinweise	Industrielle Standardverfahren können verwendet werden.
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0002:01`
Titel	**Feedback-Protokoll**
Beschreibung	Nach Auslösung durch Tf wird eine Signalisierung als Feedback zurückgegeben, indem der Zustand des Systems und Status übermittelt werden.
Hinweise	Industrielle Standardverfahren können verwendet werden.
Bewertung	Nicht erfüllt. Keine Eigendiagnosefunktion für Use Case 1 umgesetzt. Das Hauptaugenmerk liegt in der Demonstration der *Asset Identity* und der VCs mit DID.

3.2.1.11 Szenario *Blockchain Recorder*: Validität prüfen, Daten aufzeichnen – verpflichtend

Kurzbeschreibung	Die Signatur der übertragenen Daten wird mit dem öffentlichen Schlüssel des Assets auf Gültigkeit geprüft. Nach positiver Prüfung werden die übertragenen Daten im BR gespeichert.
Akteure	—
Vorbedingungen	Öffentlicher Schlüssel des Assets muss vorhanden sein.
Fachlicher Auslöser	—
Hauptszenario	Allgemeingültige Szenario
Ergebnis	Manipulationssichere Aufzeichnung der Daten von Assets

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0010:01`
Titel	**PKI-Verfahren**
Beschreibung	Für die Prüfung der Signatur ist der öffentliche Schlüssel des Assets erforderlich.
Hinweise	—
Bewertung	Erfüllt Der öffentliche Schlüssel ist über das *IDunion*-Netzwerk verfügbar.

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0011:01`
Titel	**Speichern der Daten im BR**
Beschreibung	Manipulationssichere Speicherung der Daten im BR
Hinweise	—
Bewertung	Erfüllt im Rahmen des Use Case 2 und Use Case 3

3.2.1.12 Szenario Zug herunterfahren – optional

Kurzbeschreibung	Einleitung eines kontrollierten Herunterfahrens. Signalisierung aller kritischer Zugelemente um eine Fehldiagnose zu vermeiden.
Akteure	Tf und Zugsteuerung
Vorbedingungen	Das System ist vollständig integriert und liegt betriebsbereit vor, der Zug befindet sich nicht in einer Fahrt.
Fachlicher Auslöser	Tf
Hauptszenario	Zug befindet sich im Stillstand.
Ergebnis	Zug ist ausgeschaltet.

3.2.1.13 Szenario Zugsteuerung: Herunterfahren der Assets nach Register – optional

Kurzbeschreibung	Nach Empfang des Befehls der Zugsteuerung zum Herunterfahren werden keine Sensordaten mehr an Zugsteuerung geliefert und die Assets schalten sich in den Schlafmodus
Akteure	Zugsteuerung und Assets
Vorbedingungen	Siehe Unterabschnitt 3.2.1.12
Fachlicher Auslöser	Zugsteuerung
Hauptszenario	Siehe Unterabschnitt 3.2.1.12
Ergebnis	Alle Assets sind im Schlafmodus

3.2.1.14 Szenario Asset nach Betrieb

Siehe Abbildung 3.9.

3.2.1.15 Szenario *Asset Life Cycle* (Nach Betrieb) – verpflichtend

Kurzbeschreibung	Das Asset wird aus einem Umfeld ausgebaut und deregistriert.
Akteure	Besitzer
Vorbedingungen	Das Asset ist nicht mehr funktionsfähig und muss ausgebaut werden.

Fachlicher Auslöser	Ausbau eines Assets aus einem dafür vorgesehenen Betriebsumfeld (z. B. Zug- oder Service-Station)
Hauptszenario	—
Ergebnis	Das Asset ist von der Zugsteuerung abgemeldet und physikalisch ausgebaut. Zusätzlich ist ein Zertifikat für die nächste Verwendung oder dem Recycling ausgestellt worden.

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0003:01`
Titel	**Ausbau des Assets**
Beschreibung	—
Hinweise	—
Bewertung	Erfüllt Siehe Prozess Wartung

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetNutzung_#DEF0003:01`
Titel	**Nächste Verwendung**
Beschreibung	—
Hinweise	—
Bewertung	Erfüllt Siehe Prozess Wartung

Asset-Historie Die Asset-Historie wird in drei Schritten festgehalten. Diese unterteilen sich in vor, während und nach dem Betrieb. Die statischen Daten des Assets werden vor allem durch Zertifikate (Credentials) entlang der Wertschöpfungskette widergespiegelt. Diese werden nicht in der BR auf dem Zug gespeichert. Das Konzept der DIDs und VCs wird demnach nur in der Cloud für die statischen Daten verwendet. Die dynamischen Daten, die für die BR auf dem Zug von den Assets gespeichert werden, werden durch einfache Schlüsselpaare auf dem Asset unterschrieben. Erst nachdem der Speicher auf dem Zug voll ist, werden die dynamischen Daten in die Cloud exportiert. Diese Daten sind durch den Mechanismus der Wallet-Erstellung (siehe Szenario „Asset erstellen“) verifizierbar und zuzuordnen zu dessen DID und dessen statischen Daten (VCs.).

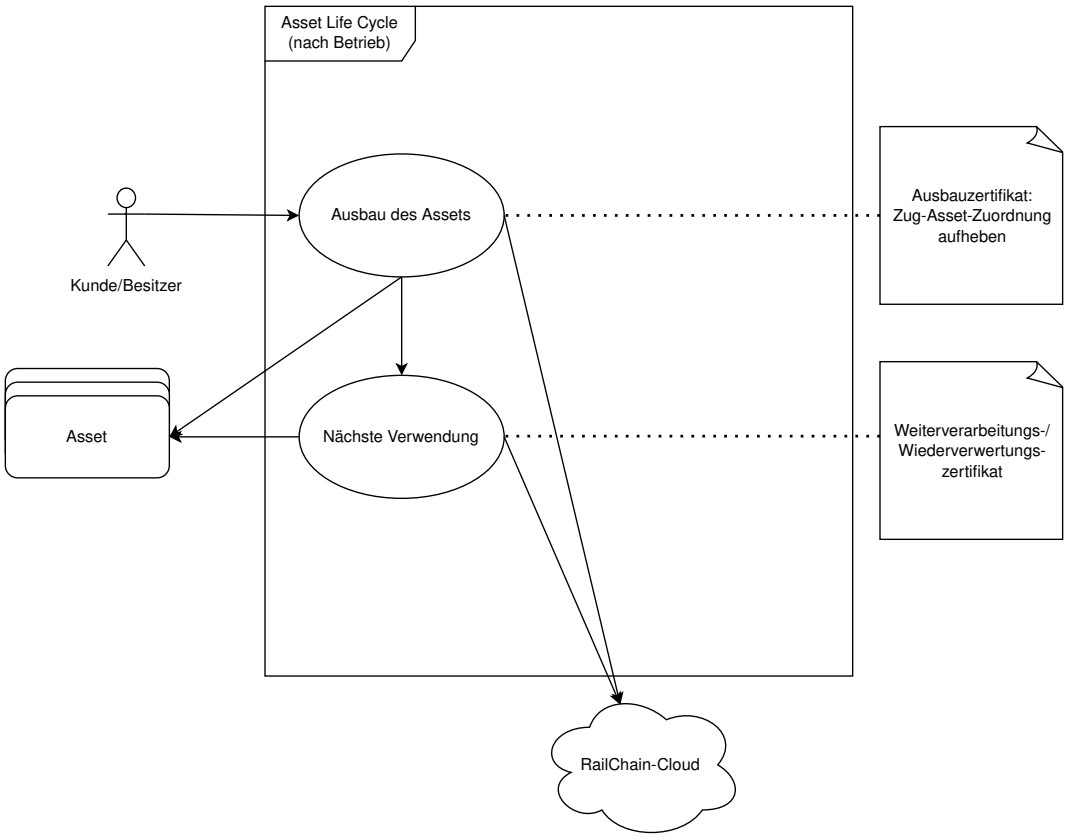

Abbildung 3.9: *Asset Life Cycle* III

3.2.1.16 Szenario Zug Zwischenspeicher auslesen und permanent speichern – optional

Kurzbeschreibung	Der Zwischenspeicher der Zugsteuerung wird als Momentaufnahme erfasst und in eine Datenbank/Blockchain permanent gespeichert. Im Anschluss wird der Zwischenspeicher geleert.
Akteure	—
Vorbedingungen	Es existiert eine Datenbank/Blockchain lokal im Zug. Gegebenenfalls Übertragung der Historie in die DB-Cloud.
Fachlicher Auslöser	Übertragung der Historie erfolgt: • während der Fahrt in zyklischen Abständen • vor geplanten Instandhaltungen

Hauptszenario	Während der Fahrt werden Sensordaten als auch Inbetriebnahmedaten (Self-Check) in den Zwischenspeicher übertragen. Zu festgelegten Zeitpunkten wird der gesamte Inhalt des Zwischenspeichers als Historie permanent erfasst. Die Historie kann eindeutig über die Asset-ID zugeordnet werden.
Ergebnis	Vollständige Verfügbarkeit der Historie eines Assets. Beginnend mit der ersten Inbetriebnahme bis zum Ende des Lebenszyklus des Assets (inkl. Besitzerwechsel des Assets).

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetHistorie_#DEF0001:01`
Titel	**Auslesen des Zwischenspeichers im Zug**
Beschreibung	Zwischenspeicher des Zuges beinhaltet Datensätze, die zu jedem Zeitpunkt auslesbar sind.
Hinweise	—
Bewertung	Nicht erfüllt. Wegen der Rückwirkungsfreiheit konnte diese Anforderung nicht umgesetzt werden.

Identifikator	`#REQ-RC_UC1_AssetHistorie_#DEF0002:01`
Titel	**Dauerhaftes Speichern**
Beschreibung	Die vom Zwischenspeicher ausgelesenen Datensätze werden in die Datenbank/Blockchain/DB-Cloud übertragend.
Hinweise	—
Bewertung	Nicht erfüllt. Wegen der Rückwirkungsfreiheit konnte diese Anforderung nicht umgesetzt werden.

Identifikator	`#REQ-RC_UC1_AssetHistorie_#DEF0003:01`
Titel	**Löschen des Zwischenspeichers**
Beschreibung	Nach dem permanenten Speichern der Datensätze in die Datenbank/Blockchain/DB-Cloud wird der Zwischenspeicher geleert.
Hinweise	Nur nach dem erfolgreichen Speichern erfolgt dieser Vorgang.
Bewertung	Nicht erfüllt. Wegen der Rückwirkungsfreiheit konnte diese Anforderung nicht umgesetzt werden.

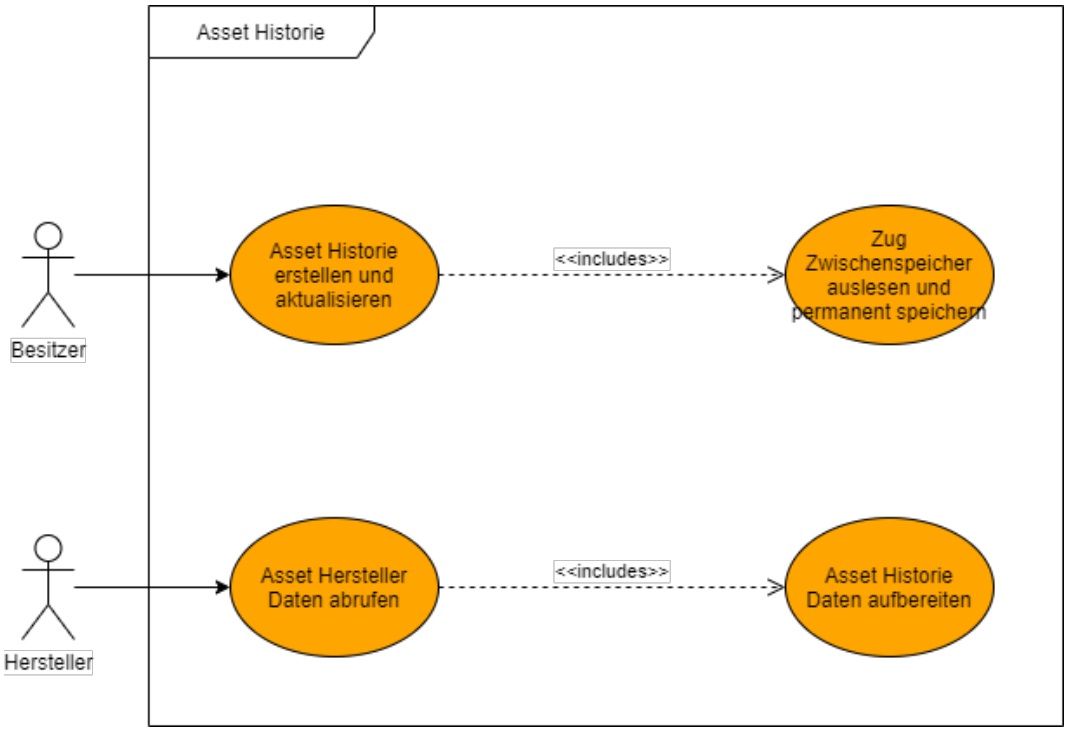

Abbildung 3.10: Asset-Historie

3.2.1.17 Szenario Asset-Hersteller: Daten abrufen – verpflichtend

Kurzbeschreibung	Alle gespeicherten Daten (z. B. Ein- und Ausbau, Besitzerwechsel, usw.), die zu dem Asset gehören und Datensätze des *Internet of Things* (IoT) sind bereits erfasst (Blockchain/Datenbank). Über eine Rest-API-Schnittstelle können die Daten abgerufen werden. Der Schlüsselparameter ist in diesem Fall immer das DID-Dokument.
Akteure	Anfragesteller (Hersteller)
Vorbedingungen	Es wurden bereits Datensätze erfasst.
Fachlicher Auslöser	—
Hauptszenario	Ein Hersteller kann über das DID-Dokument die Historie abrufen.
Ergebnis	—

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetHistorie_#DEF0004:01`
Titel	**Schnittstelle zur Historie**
Beschreibung	Schnittstelle für den Zugriff auf den Historienspeicher

Hinweise	—
Bewertung	Erfüllt Siehe Prozess Herstellerzertifikat

Identifikator	`#REQ-RC_UC1_AssetHistorie_#DEF0005:01`
Titel	**Datenablage**
Beschreibung	Die erfassten Daten werden in einer Datenbank oder Blockchain erfasst. Zu diesem Historiendatensatz existiert immer ein DID-Dokument als Referenz.
Hinweise	—
Bewertung	Erfüllt Siehe Prozess Herstellerzertifikat

3.2.1.18 Szenario Asset-Historie Daten aufbereiten – optional

Kurzbeschreibung	Alle erfassten Historiendaten zu einem Asset können über eine Filterfunktion angepasst werden.
Akteure	Besitzer und Anfragesteller (Hersteller)
Vorbedingungen	Es liegt eine Anfrage an dem Asset vor.
Fachlicher Auslöser	Hersteller
Hauptszenario	Über eine Filterfunktion ist es möglich, den Historienverlauf einzuschränken. Dazu ist der Besitzer des Assets berechtigt. Bei einem Besitzerwechsel kann dieser nicht mehr die Filterfunktion verändern. Dadurch wird gewährleistet, dass nach einem Besitzerwechsel der neue Besitzer individuelle betriebsrelevante Informationen nicht nachvollziehen kann. Es liegt ein angepasster Historienverlauf des Assets vor.

Anforderungen

Identifikator	`#REQ-RC_UC1_AssetHistorie_#DEF0006:01`
Titel	**Filterfunktion**
Beschreibung	Die Filterfunktion berechtigt den Besitzer, seine Historiendaten individuell anzupassen. Das heißt, Felder komplett auszublenden, die nur von ihm einsehbar sind.
Hinweise	Die Filterfunktion kann nur vom aktuellen Besitzer des Assets verwendet werden. Bei einem Besitzerwechsel ist der Zugriff nicht mehr möglich.
Bewertung	Nicht erfüllt.

3.3 Use Case 2: Datenaufzeichnung ohne Echtzeitanforderungen

Dieser Use Case beschreibt die Anforderungen an das Projekt *RailChain* und welche Problemstellungen gelöst werden müssen, um den Einsatz der DLT in einem Zug sinnvoll zu gewährleisten.

Eine zuverlässige Speicherfunktion von möglichst vielen Informationen wird allein durch die massiv fortschreitende Digitalisierung sämtlicher technischen Vorgänge innerhalb eines Zuges notwendig, um Nachvollziehbarkeit, Transparenz und Auditierbarkeit auch zukünftig beherrschen zu können, ohne den Anspruch an den Datenschutz aus den Augen zu verlieren.

Die Speicherung von Zuginformationen in einer JRU dient heute zur Unterstützung bei der Unfallanalyse mit hohen Anforderungen an Unveränderbarkeit und Unversehrtheit. Diese hohen Anforderungen führen zu technischen Lösungen mit sehr restriktiven Einschränkungen für Kapazität und Zugänglichkeit. Zum anderen existieren im Zug herstellerspezifische Diagnosesysteme für die Aufzeichnung von technischen Informationen zu Wartungs- und Analysezwecken.

Die DLT kombiniert *Peer-to-Peer*-Netzwerke mit starker Kryptographie, um auf den Netzwerkknoten einen gleichen und gesicherten Datenbestand zu gewährleisten, dessen Integrität im Vergleich zu normalen IT-Systemen nur sehr schwer zu kompromittieren ist. Die grundsätzliche Idee ist, unter Ausnutzung dieser neuen Technologie, die Voraussetzungen zu schaffen, mit konventioneller Hardware die Anforderungen an eine Datenaufzeichnung im Zug zu erfüllen, die sowohl für technische Diagnosesysteme als auch für die JRU genügen.

Die Struktur der Use Cases ist bewusst so aufgeteilt, dass die Prüfung für den Anwendungsfall JRU als letzte „Königsdisziplin“ in Use Case 3 verlagert wird, da die Komplexität der Technik ein einfacheres Szenario erfordert. Selbst wenn das Projekt im Use Case 3 als Ergebnis zu dem Schluss kommt, dass die Anforderungen einer JRU (noch) nicht erfüllbar sind, dann hätten die Ergebnisse für zukünftige Anwendungen weiterhin Relevanz.

Das Projekt leistet durch den Einsatz eines völlig neuen technologischen Ansatzes Pionierarbeit in einem stark regulierten Umfeld, da die nutzenstiftenden Potenziale von enormer Bedeutung für die Flexibilisierung und Standardisierung im Rahmen der Digitalisierung sind.

3.3.1 Funktionale Anforderungen

Bei dem Use-Case-Diagramm liegt das Hauptaugenmerk in der Funktionalität des Systems aus Sicht des Nutzers. Das System erfüllt den wesentlichen Nutzen, Daten aus dem Zugbus (s. Kapitel 2) aufzuzeichnen und bei Bedarf bereitzustellen. Im folgenden Use-Case-Diagramm sind die Grundfunktionen dargestellt:

Anforderungen eines Use Cases sollten normalerweise technologie-agnostisch sein. Die Anforderungen an derzeit verwendete Aufzeichnungssysteme sind im Detail allerdings auch stark an die verwendeten Technologien angelehnt. Die neuen technologischen Möglichkeiten der DLT (verteilte Architektur ggü. Einzelgerät)

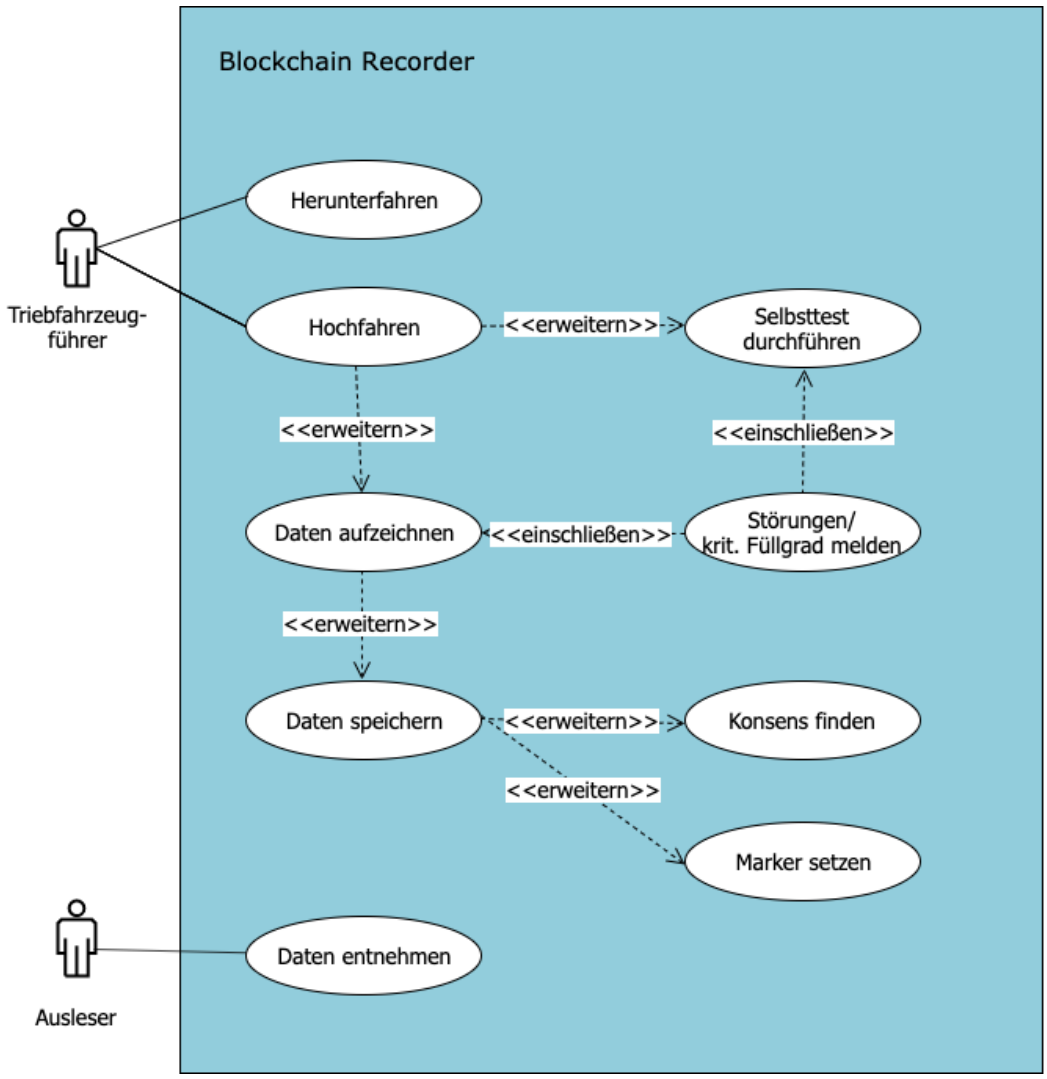

Abbildung 3.11: Use-Case-Diagramm Datenschreiber

erfordern anders geartete Anforderungen an die Systeme. In den folgenden funktionalen Anforderungen sind daher technologiebezogene Details vorhanden, um eine Vergleichbarkeit mit derzeit angewendeten Technologien zu erhalten.

3.3.1.1 Szenario Hochfahren

Name	**Hochfahren**
Kurzbeschreibung	Der Tf schaltet den Zug ein. Betriebsbereitschaft der Assets wird hergestellt.
Akteure	Tf
Vorbedingungen	keine
Fachlicher Auslöser	Zug in Betriebsbereitschaft setzen.
Ergebnis	Zug und Assets sind betriebsbereit.
Hauptszenario	Beim Einschalten des Zuges werden folgende Schritte durchgeführt: 1. Stromversorgung für alle Geräte liegt an. 2. Geräte führen einen Selbsttest durch. 3. Buskommunikation wird initialisiert; der Zustand aller Assets wird über die Businitialisierung erfasst; die Datenerfassung über den Zugbus muss hier bereits funktionieren, weil die Initialisierung für die korrekte Interpretation wichtige Informationen enthält. 4. Alle Assets sind fehlerfrei hochgefahren.
Alternativszenarien	—
Nachbedingungen	Zug ist betriebsbereit.

Identifikator	`#REQ-RC_UC2_Hochfahren_#DEF0001:01`
Titel	**Stromversorgung und Synchronisierung mit dem Zugbus**
Beschreibung	Die Initialisierung der BR-Knoten muss so abgestimmt sein, dass eine Aufzeichnung von Zustandsdaten vom Zugbus lückenlos erfolgt.
Ursprung	—
Querbezüge	—

Bewertung	Erfüllt Die Stromversorgung ist an den Batterieschalter gekoppelt, d. h. mit der ersten Stufe des Hochfahrens des Zuges fahren die BR-Knoten hoch. Die Synchronisierung mit dem Zugbus erfolgt sobald die BR-Knoten vollständig hochgefahren sind. Das Hochfahren des Zuges konnte ohne die Zugzulassung zu verlieren nicht so gestaltet werden, dass jede Information vom MVB auch im BR gespeichert wird. Für die Erfüllung als BR und JBR ist das als ausreichend angesehen worden, da die Systeme innerhalb von 40 Sekunden vollständig synchronisiert und arbeitsbereit waren. Sollte es Use Cases geben, die auf diese Zeitspanne spezialisiert Daten auswerten müssen, so sind dafür spezielle Vorkehrungen zu treffen, die eine Aktivierung des Zuges von der Bereitschaft der BR-Knoten abhängig machen.

Identifikator	`#REQ-RC_UC2_Hochfahren_#DEF0002:01`
Titel	**Initialisierung des BR (Versuchsträger)**
Beschreibung	Der BR prüft die Konsistenz des lokalen Speicherplatzes und gleicht diesen über Verbindungen zu anderen Knoten ab. Ziel: Aufzeichnungsfähigkeit erzeugen
Ursprung	—
Querbezüge	—
Identifikator	`#REQ-RC_UC2_Hochfahren_#DEF0002:01`
Titel	**Initialisierung des BR (Versuchsträger)**
Beschreibung	Konsistenz des Speicherplatzes prüfen, Verbindungen zu anderen Knoten aufsetzen, Bearbeitungsstand abgleichen Ziel: Aufzeichnungsfähigkeit erzeugen Die nachfolgenden Anforderungen sind auch nach der erreichten Aufzeichnungsfähigkeit durchführbar: • Verbindung zur *RailChain*-Cloud prüfen • Hochladen in *RailChain*-Cloud prüfen – siehe „Fahrdaten hochladen“
Ursprung	—
Querbezüge	—

Bewertung	Erfüllt Im Laufe des Projektes wurde entschieden, die Initialisierung der Blockchain beim Hochfahren des Zuges zu initialisieren. Die vorher angelegte Blockkette bleibt erhalten und ist in sich konsistent und kann jederzeit durch die LTE-Datenverbindung abgerufen werden. Es ist demnach keine Synchronisation mit dem Datenbestand vor der Inbetriebnahme notwendig. Aus den Inhalten der gespeicherten Daten kann nachvollzogen werden, ob die Zugdaten in sich konsistent sind oder ob es Lücken in der Aufzeichnung gegeben hat.

Identifikator	`#REQ-RC_UC2_Hochfahren_#DEF0003:01`
Titel	**Aufzeichnungsstand mit *RailChain*-Cloud abgleichen**
Beschreibung	Der BR prüft die Verbindung zur *RailChain*-Cloud, ggf. wird ein Hochladen durchgeführt: siehe „Fahrdaten hochladen“.
Ursprung	—
Querbezüge	—
Bewertung	Nicht relevant. Durch die Entscheidung beim Hochfahren eine Neuinitialisierung der Blockchain durchzuführen ist ein Abgleich nicht mehr nötig.

3.3.1.2 Szenario Daten aufzeichnen

Name	**Daten aufzeichnen**
Kurzbeschreibung	Der BR registriert die relevanten Fahrdaten, die über den Zugbus an ihn kommuniziert werden.
Akteure	Fahrzeugbediener
Vorbedingungen	Der BR ist aufzeichnungsbereit
Fachlicher Auslöser	Aufzuzeichnende Nachrichten werden vom BR als solche erkannt.
Ergebnis	Daten sind im Speichermedium des BR gespeichert.
Hauptszenario	Auf dem Fahrzeugbus gesendete Nachrichten werden vom BR erkannt. Dabei werden sie ihrem Inhalt nach klassifiziert und in entsprechenden Speicherbereichen abgelegt.
Alternativszenarien	—
Nachbedingungen	Der BR ist rechtzeitig bereit, weitere Nachrichten zu erkennen und abzuspeichern.

Anforderungen

Identifikator	`#REQ-RC_UC2_DatenAufzeichnen_#DEF0001:01`
Titel	**Schnittstellen zu den Fahrdaten**
Beschreibung	Der BR besitzt eine Schnittstelle zum Zugbus. Hinweis: Der BR bezieht im Projekt *RailChain* seine Daten ausschließlich vom Zugbus. Die Anforderungen an den Datenrecorder beinhalten viele andere technische Schnittstellen. Diese Schnittstellen sind im Rahmen dieses Projekts nicht erforderlich um die Systemkomplexität nicht weiter zu erhöhen.
Ursprung	—
Querbezüge	—
Bewertung	Erfüllt Die Schnittstelle zu den Fahrdaten wird durch einen Datenabgriff auf den MVB realisiert. Die DB Systemtechnik hat ein Telematiksystem bereitgestellt, das diesen Tatbestand erfüllt und hat das Projekt aktiv dabei unterstützt die Auswahl und Aufbereitung der aufzuzeichnenden Fahrdaten für die BR-Knoten umzusetzen. Neben den für den JBR-Betrieb relevanten ca. 70 Werten ist es dem Projekt gelungen bis zu 800 Werte vom MVB aufzugreifen und im BR zu speichern.

Identifikator	`#REQ-RC_UC2_DatenAufzeichnen_#DEF0002:01`
Titel	**Rückwirkungsfreiheit Aufzeichnungssystem (Versuchsträger)**
Beschreibung	Der Versuchsausbau muss rückwirkungsfrei für den Zugbetrieb sein. Die Rückwirkungsfreiheit muss durch entsprechende Gutachten nachgewiesen sein. Hinweis: Ohne Rückwirkungsfreiheit ist die Betriebsgenehmigung für den Zug nicht gefährdet.
Ursprung	—
Querbezüge	Deckungsgleich mit nicht-funktionalen Anforderungen Rückwirkungsfreiheit
Bewertung	Erfüllt Dem Aufbau der Telematik-Box durch die DB Systemtechnik ist eine Rückwirkungsfreiheit per Gutachten verifiziert worden.

Identifikator	`#REQ-RC_UC2_DatenAufzeichnen_#DEF0003:01`
Titel	**Durchsatz des JBR**
Beschreibung	Der MVB verfügt über den Durchsatz von 1,5 Mbit/s und Nutzlast von 851 kbit/s. Dabei wird im maximalen Durchsatzfall der maximal praktikable Durchsatz oder technisch machbare Durchsatz angenommen. Die Ergebnisse des Projektes sollen auch Grenzen der Aufzeichnungskapazität aufzeigen, die in Relation zu den Kapazitäten von moderneren Zugbus-Systemen (*ProfiBus*, OCORA [3]) stehen. Die Mindestgrenze sind die Werte, die heutzutage durch eine JRU aufgezeichnet werden.
Ursprung	—
Querbezüge	—
Hinweise	Dabei zu beachten sind die Grenzen, bis das 1-Sekunden-Kriterium verletzt wird.
Bewertung	Erfüllt Der JBR im Zug ist nachgewiesenermaßen in der Lage mehr als 800 Werte aus dem MVB lückenlos aufzuzeichnen. Dabei wird die Konsenszeit noch unter 0,1 Sekunden erreicht. Bei Bedarf ist es durchaus möglich höhere Datendurchsätze zu erreichen.

Identifikator	`#REQ-RC_UC2_DatenAufzeichnen_#DEF0004:01`
Titel	**Die Datenaufzeichnung hat alle interpretierbaren Daten aufzuzeichnen**
Beschreibung	Welche Daten konkret aufgezeichnet werden ist im Kontext dieser Anforderungen nicht von vordergründiger Bedeutung. Vielmehr sind die Menge der anfallenden Daten durch den Zugbus und das dazu erfahrbare Zeitverhalten für den Konsensmechanismus Gegenstand der Betrachtungen. Die Ergebnisse des Projektes sollen abhängig von der Parametrierung des Versuchsaufbaus und Variierung der Zugbus-Auslastung das Zeitverhalten des Konsensmechanismus aufzeigen.
Ursprung	—
Querbezüge	Parametrierung des Versuchsaufbaus s. Daten speichern
Hinweise	—
Bewertung	Erfüllt Siehe vorhergehende Anforderungen

Identifikator	`#REQ-RC_UC2_DatenAufzeichnen_#DEF0005:01`
Titel	**Zeitstempel**
Beschreibung	Der BR muss die aufzuzeichnenden Nachrichten mit einem über den Fahrzeugbus synchronisierten, einheitlichen Zeitstempel nach UTC-Zeit und Datum versehen.
Ursprung	—
Querbezüge	—
Bewertung	Erfüllt Jeder Datensatz ist mit einem UTC-Zeitstempel versehen

Identifikator	`#REQ-RC_UC2_DatenAufzeichnen_#DEF0006:01`
Titel	**Ortsstempel**
Beschreibung	Der BR muss die aufzuzeichnenden Nachrichten mit einer Angabe über den Ort der Entstehung versehen.
Ursprung	—
Querbezüge	—
Bewertung	Erfüllt Neben den MVB-Daten wird durch eine GPS-Antenne auf dem Dach des Zuges das GPS-Signal mit dem entsprechende UTC-Zeitstempel gespeichert.

3.3.1.3 Szenario Daten speichern

Name	**Daten speichern**
Kurzbeschreibung	Dieses Szenario formuliert die Anforderungen an die Aufzeichnung der Daten. Hier werden die Kernanforderungen an die DLT gestellt und die wesentlichen Aspekte der Eignung der Technologie beschrieben.
Akteure	Zugbus, BR
Vorbedingungen	Der BR ist aufzeichnungsbereit und der Zugbus ist initialisiert.
Fachlicher Auslöser	Aufzuzeichnende Daten liegen am Zugbus an
Ergebnis	Daten sind im Speichermedium des BR gespeichert und Konsens zwischen der Mehrheit der BR-Knoten ist erreicht.
Hauptszenario	Auf dem Fahrzeugbus gesendete Nachrichten werden vom BR erkannt und aufgezeichnet.
Alternativszenarien	—
Nachbedingungen	—

Identifikator	`#REQ-RC_UC2_DatenSpeichern_#DEF0001:01`
Titel	**Framework des BR**
Beschreibung	Ein Framework (Standard wie z. B. Quorum oder HLF, oder Eigenentwicklung) als eine Grundstruktur für den BR soll eingesetzt werden und entsprechend den Anforderungen vom BR konfiguriert werden. Wichtige Kriterien: • Skalierbarkeit • Leistung • Konsensmechanismus • Konsenszeit • Ausfallsicherheit • Fälschungssicherheit • Grad der Standardisierung • Robustheit
Ursprung	—
Querbezüge	—
Bewertung	Erfüllt Die bekannten Blockchain-Frameworks haben sich nach kurzer Überprüfung alle als zu schwerfällig und zu ressourcenhungrig herausgestellt, um in einem realen Zugszenario eingesetzt zu werden. Zudem sind transaktionsbasierte oder darauf aufbauende Funktionen (z. B. *Smart Contracts*) nicht erforderlich. Die Kernfunktionen für die fälschungssichere Datenaufzeichnung sind die Block- und Konsensbildungsfähigkeiten. Die TU Braunschweig hat durch Ihre Vorerfahrung auf das Themis-Framework [49] zurückgreifen können und durch Labortests die hervorragende Eignung unter Beweis stellen können.

Identifikator	`#REQ-RC_UC2_DatenSpeichern_#DEF0002:01`
Titel	***On-chain* aufzuzeichnende Daten**
Beschreibung	Es muss festgelegt werden, welche Daten in der Blockchain aufgezeichnet werden: Die Fahrdaten (sog. Speicherung *on-chain*) oder nur die Hashwerte von Fahrdaten (sog. Speicherung *off-chain*). Auswahlkriterien: • Durchsatz
Ursprung	—
Querbezüge	—
Hinweise	Datenschutz (personenbezogene Daten)
Bewertung	Erfüllt Die wesentliche Anforderung der Datenaufzeichung im BR ist die Fälschungssicherheit. Das gewählte Systemdesign hat es ermöglicht, alle aufzuzeichnenden Daten *on-chain* gestalten zu können.

Identifikator	`#REQ-RC_UC2_DatenSpeichern_#DEF0003:01`
Titel	**Kontinuierliche Aufzeichnung**
Beschreibung	Wenn der BR im Aufzeichnungsmodus ist, muss er eingehende Daten kontinuierlich speichern.
Ursprung	—
Querbezüge	—
Hinweise	—
Bewertung	Erfüllt Die über die letzten 1,5 Jahre des Projektes verlaufende Dauertestphase hat eindeutig bewiesen, dass mehr als die geforderten Daten kontinuierlich im Rahmen der zeitkritischen Grenzen vom System aufgezeichnet werden konnten und deren Integrität auch lange nach der externen Speicherung nachgewiesen werden kann.

Identifikator	`#REQ-RC_UC2_DatenSpeichern_#DEF0005:01`
Titel	**Verschlüsselung Personendaten**
Beschreibung	Personenbezogene Daten werden im Fahrtenschreiber verschlüsselt gespeichert.
Ursprung	DB_LH_JRU Kapitel 2 (1); DB_LH_JRU Kapitel 2 (3);
Querbezüge	`#REQ-DB_LH_Security-0002:01#DEF#`

Hinweise	Einhaltung der Datenschutzgesetze: Der Zugriff auf personenbezogene Daten soll nur einem bestimmten Personenkreis zugänglich gemacht werden. Als personenbezogene Daten zählt beispielsweise Nummer des Tf
Bewertung	Nicht erfüllt, jedoch leicht lösbar im Falle einer konkreten industriellen Anwendung. Die Daten werden zunächst ohne Verschlüsselung in die Blockchain geschrieben. Die personenbezogenen Daten sind nicht direkt interpretierbar, sondern müssten erst durch eine Referenztabelle aufgelöst werden. Eine mögliche Lösung ist diese Daten zusätzlich zu verschlüsseln bevor sie gespeichert werden oder den Zugang wie bei klassischen Systemen durch Zugangsbarrieren und Authentifizierungsmechanismen zu verhindern.

Identifikator	`#REQ-RC_UC2_DatenSpeichern_#DEF0006:01`
Titel	**Verschlüsselung Fahrdaten**
Beschreibung	Fahrdaten und personenbezogene Daten dürfen nicht denselben Schlüssel haben.
Ursprung	—
Querbezüge	`#REQ-DB_LH_Security-0002:01#DEF#`
Hinweise	—
Bewertung	Nicht erfüllt, jedoch leicht lösbar im Falle einer konkreten industriellen Anwendung. s. vorhergehende Anforderung

Identifikator	`#REQ-RC_UC2_DatenSpeichern_#DEF0007:01`
Titel	**Signierung**
Beschreibung	Die gespeicherten Fahrdaten müssen signiert werden.
Ursprung	IEC_62625 Kapitel 4.2.5.5
Querbezüge	—
Hinweise	—
Bewertung	Erfüllt Die Daten werden zwar nicht einzeln mit einer Signatur versehen, aber jeder gespeicherte Block erhält eine Signatur, die im nächsten Block abgelegt wird.

Identifikator	`#REQ-RC_UC2_Schutzspeichern_#DEF0008:01`
Titel	**Aktiver Schutz**
Beschreibung	Der BR muss die Daten vor Verlust oder Beschädigung schützen.
Ursprung	—
Querbezüge	`#REQ-DB_LH_Security-0004:01#DEF#`
Hinweise	Nicht über das Standardmaß hinaus, das für andere IT-Komponenten gilt (Use Case 3 kann jedoch strengere Kriterien fordern).
Bewertung	Erfüllt Die Daten werden im Versuchsaufbau über vier Knoten über den Konsensmechanismus konsistent gespeichert. Jeder Knoten erhält durch den Abgleich der Signaturen auch die Information, dass alle anderen Knoten den gleichen Informationsstand haben. Sollte der Bedarf für den Datenverlust noch höher sein, so kann die Anzahl der Knoten beliebig erweitert werden.

Identifikator	`#REQ-RC_UC2_Datenspeichern_#DEF0009:01`
Titel	**Fälschungssichere Speicherung**
Beschreibung	Der BR muss die gespeicherten Fahrdaten fälschungssicher abspeichern.
Ursprung	—
Querbezüge	`#REQ-DB_LH_Security-0003:01#DEF#`
Hinweise	—
Bewertung	Erfüllt Die Fälschungssicherheit ergibt sich aus der Bildung der Blockchain und der Signatur jedes vorhergehenden Blockes. Die öffentlichen Schlüssel sind in den Kopfzeilen der Blockkette enthalten. Die privaten Schlüssel zur Erzeugung der Signaturen müssten von allen BR-Knoten gleichzeitig kompromittiert werden. In Anwendungen mit industriellem Maßstab ist es ratsam, für den Schutz der privaten Schlüssel auf HSE-Komponenten zurückzugreifen.

3.3.1.4 Szenario Daten entnehmen

Der BR soll die Verfügbarkeit der aufgezeichneten Daten durch deren erleichtertes Hochladen erhöhen.

Name	**Daten entnehmen**
Kurzbeschreibung	Die Daten werden über eine WLAN-Schnittstelle an einen übergeordneten Speicher übertragen.
Akteure	Das Hochladen der Daten wird entweder manuell durch den Ausleser oder automatisch mit der Anmeldung der Hochladestation ausgelöst.
Vorbedingungen	Der BR befindet sich stationär an einem zum Hochladen geeigneten Ort (z. B. zur Übernachtung im Depot).
Fachlicher Auslöser	Der BR erhält einen berechtigten Befehl zum Hochladen.
Ergebnis	—
Hauptszenario	Es wird eine Netzwerkverbindung zu einem autorisierten Empfänger hergestellt, der das Hochladen durch einen berechtigten Befehl anstößt. Nach durchgeführtem Hochladen wird der Füllstand des BR zurückgesetzt und ein Marker gesetzt.
Alternativszenarien	Alternativ wird das Hochladen nicht automatisiert infrastrukturseitig, sondern manuell von berechtigtem Bedienpersonal (Ausleser) ausgelöst. Dies geschieht entweder durch einen tragbaren Empfänger (z. B. durch Wartungspersonal) oder im Falle eines beschädigten BR durch direkten Zugriff auf das Speichermedium.
Nachbedingungen	Der BR ist dazu bereit einen neuen Hochladebefehl zu empfangen oder Fahrdaten aufzuzeichnen.

Identifikator	`#REQ-RC_UC2_Datenentnahme_#DEF0001:01`
Titel	**Verfälschungssicherheit**
Beschreibung	Bei der Übertragung der elektronischen Fahrdaten gewährleistet eine Datensicherungsschicht eine fälschungssichere Übertragung.
Ursprung	—
Querbezüge	`#REQ-DB_LH_JRU-Schnittstelle_Out -0002:01#DEF#`, `#REQ-DB_LH_Security-0003:01#DEF#`
Hinweise	—

Bewertung	Erfüllt Die Datenentnahme erfolgt über eine Tunnelverbindung, die durch starke Authentifizierungs- und Verschlüsselungstechnologie (ssh mit 256bit ed25519 Schlüssel) geschützt ist. Weiterhin wird die Blockchain nach dem Herunterladen auf der Empfängerseite sofort auf Integrität überprüft, bevor der Marker auf dem BR für die Datenentnahme gesetzt wird.

Datenzugriff Der Zugriff auf die gespeicherten Fahrdaten soll nur einem autorisierten Personenkreis möglich sein. Besonders die personenbezogenen Daten unterliegen aus datenschutzrechtlichen Gründen besonderen Anforderungen.

Identifikator	`#REQ-RC_UC2_Datenzugriff_#DEF0001:01`
Titel	**Speicherzugang**
Beschreibung	Der Zugang zum Speicher des BR muss so gesichert sein, dass nur autorisiertem Personal der Zugriff möglich ist.
Ursprung	—
Querbezüge	—
Hinweise	—
Bewertung	Nicht erfüllt, jedoch leicht lösbar im Falle einer konkreten industriellen Anwendung. s. `#REQ-RC_UC2_DatenSpeichern_#DEF0005:01`

Identifikator	`#REQ-RC_UC2_Datenzugriff_#DEF0002:01`
Titel	**Speichermanipulation**
Beschreibung	Der Zugang zum Speicher des BR muss so gesichert sein, dass eine Manipulation der Rohdaten technisch ausgeschlossen ist.
Ursprung	—
Querbezüge	—
Hinweise	Dies betrifft das Schreiben, Modifizieren und Löschen von Fahrdaten
Bewertung	Erfüllt Sobald die Daten in der Blockchain abgelegt sind würde eine Manipulation nachvollziehbar sein. Eine unbemerkte Manipulation müsste auf allen Knoten gleichzeitig erfolgen und auf den heruntergeladenen Cloud-Speichern ebenfalls.

Identifikator	`#REQ-RC_UC2_Datenzugriff_#DEF0003:01`
Titel	**Überschreiben verhindern**
Beschreibung	Beim Hochladen der Fahrdaten muss verhindert werden, dass bereits archivierte Daten überschrieben werden.
Ursprung	—
Querbezüge	—
Hinweise	—
Bewertung	Erfüllt Der Abgleich der heruntergeladenen Daten und das Zusammenfügen der Blockchain mit der Integritätsprüfung verhindert ein inkonsistentes Abspeichern außerhalb des Zuges. Die Signaturen beziehen sich auch auf die Zeitstempel, sodass eine Verwechslung ausgeschlossen ist. Die Cloud bietet nahezu unerschöpfliche Speicherressourcen, sodass ein versehentliches Überschreiben aus Platzmangel ausgeschlossen werden kann.

Identifikator	`#REQ-RC_UC2_Datenzugriff_#DEF0004:01`
Titel	**Recording zeitgleich mit Hochladen**
Beschreibung	Wenn ein Hochladen während der Datenaufzeichnung möglich ist, dürfen die Funktionen zur Aufzeichnung von Fahrdaten nicht beeinträchtigt sein.
Ursprung	—
Querbezüge	—
Hinweise	Z. B. während manuellen Auslesens während der Zugfahrt
Bewertung	Erfüllt Die Systemressourcen sind bei gleichzeitigem Aufzeichnungs- und Hochladevorgang nicht einmal zu 10 % ausgenutzt. Dadurch ergeben sich genügend Reserven für den Ausschluss einer Gefahr durch das Hochladen von Daten.

3.3.1.5 Szenario Konsens finden

Name	**Konsens finden**
Kurzbeschreibung	Die verschiedenen BR-Knoten eines Zuges müssen neue Einträge in den verteilten Speicher gegenseitig bestätigen, damit eine Konsistenz gewährleistet werden kann.
Akteure	BR

Vorbedingungen	Es existiert eine Datenmenge, über die Einigung erlangt werden soll.
Fachlicher Auslöser	Zyklisch Vor jeder Entnahme der Fahrdaten muss sichergestellt werden, dass eine Einigung über die zu entnehmenden Daten erreicht wurde.
Ergebnis	Konsistente Speicherung der Fahrdaten Fehlererkennung durch Mehrheitsprinzip
Hauptszenario	Die einzelnen BR einigen sich zyklisch über die aufgezeichneten Daten.
Alternativszenarien	—
Nachbedingungen	Die verschiedenen BR sind sich über den Stand der aufgezeichneten Daten einig. Die Einigung wird vermerkt, sodass erkenntlich ist, ab welchem Stand der fortlaufend aufgezeichneten Daten eine Einigung erreicht ist.

Identifikator	`#REQ-RC_UC2_Konsens_#DEF0001:01`
Titel	**Aufbau des BR-Netzwerks**
Beschreibung	Das BR-Netzwerk soll aus mindestens 2, vorzugsweise mehr, Knoten bestehen.
Ursprung	—
Querbezüge	—
Hinweise	Die Kriterien für die Anzahl der nötigen Knoten im BR-Netzwerk: • Die Anzahl der Knoten im BR-Netzwerk soll festgelegt werden, sodass ein Konsens innerhalb einer Sekunde erreicht wird. • Es müssen so viele Knoten im Netzwerk existieren, dass die Kompensation des Ausfalls/Fehlverhaltens eines Knotens gewährleistet wird.

Identifikator	`#REQ-RC_UC2_Konsens_#DEF0002:01`
Titel	**Konsensmechanismus**
Beschreibung	Die verschiedenen BR-Knoten eines Zuges müssen über einen Konsensmechanismus eine Übereinstimmung zu den aufgezeichneten Daten finden.
Ursprung	—
Querbezüge	`#REQ-RC_UC2_DatenSpeichern_#DEF0001:01`

Hinweise	Eine entscheidende Frage: Wiederverwendung eines etablierten Konsensmechanismus oder Eigenentwicklung eines angepassten Konsensmechanismus. Auswahlkriterien: • Skalierbarkeit • Leistung • Konsenszeit: Konsens zwischen den BR-Knoten muss innerhalb einer Sekunde erreicht werden • Ausfallsicherheit • Fälschungssicherheit
Bewertung	Erfüllt Die Anforderungen hängen sehr stark vom verwendeten Framework ab bzw. die Austauschbarkeit des Konsensmechanismus wird durch dieses vorgegeben. Insofern wird die Erfüllung dieser Anforderung durch die Erfüllung der Kriterien für `#REQ-RC_UC2_DatenSpeichern_#DEF0001:01` vorgegeben sein.

Identifikator	`#REQ-RC_UC2_Konsens_#DEF0003:01`
Titel	**Konsenszeit**
Beschreibung	Die verschiedenen BR-Knoten eines Zuges müssen über einen Konsensalgorithmus eine Übereinstimmung zu den aufgezeichneten Daten innerhalb einer bestimmten Zeit finden. Diese Dauer sollte nachvollziehbar bleiben und sich nicht mehr als 200 % von der im Use Case 3 geforderten Konsenszeit von einer Sekunde befinden.
Ursprung	—
Querbezüge	—
Hinweise	—
Bewertung	Erfüllt Die Konsenszeit im Versuchsaufbau des Zuges ist in allen gemessenen Szenarien immer unter 0,1 Sekunden geblieben.

3.4 Use Case 3: *Juridical Blockchain Recorder* mit Echtzeitanforderungen

Im dritten Use Case geht es um den JBR, bei dem ein blockchainbasiertes und rechtssicheres Aufzeichnungsverfahren für Eisenbahnen, das ETCS (*European Train Control System*) und das digitale Stellwerk (DSTW) mit Echtzeitanforderungen zum Einsatz gebracht werden. Die Vorgaben für eine JRU sind sehr spezifisch und stellen hohe Anforderungen an Integrität und Echtzeitfähigkeit des Systems, die z. T. auch an die verwendete Technologie gebunden sind. Neben der technischen Übertragbarkeit in die Bahntechnik soll zudem auch die Wirtschaftlichkeit der Blockchain-Technologie gezeigt werden. Der Use Case 3 unterscheidet sich von Use Case 2 vor allem durch die unterschiedlichen Daten, die aufgezeichnet werden müssen, sowie die erhöhten Leistungs- und Zuverlässigkeitsanforderungen. Deswegen wird hier detailliert nur auf die geänderten Anforderungen eingegangen.

Blockchain- bzw. DLT und insbesondere *Smart Contracts* bieten bei der Aufzeichnung relevanter Daten mit hohen Anforderungen an Integrität in verteilten Infrastrukturen, wie es Eisenbahnanwendungen sind, ein hohes Potenzial. Durch die räumlich verteilte Speicherung der Fahrdaten auf verschiedenen Rechnern kann die Sicherheit gegenüber Verlust der aufgezeichneten Daten erhöht werden. Blockchain-Technologien stellen durch Konsensmechanismen und den Einsatz starker Kryptographie sicher, dass die Daten über die beteiligten Systeme hinweg konsistent sind und verhindern deren Manipulation. Insbesondere vor dem Hintergrund der fortschreitenden Fragmentierung und Privatisierung der Eisenbahnindustrie, bei denen das Vertrauen zwischen allen Parteien zukünftig nicht notwendigerweise vorhanden ist, bieten diese Technologien bedeutendes Potenzial.

3.4.1 Zielstellung Use Case 3 JBR

Das Projekt *RailChain* hat zum Ziel Ergebnisse auf Basis von praktischen Erfahrungen zu gewinnen. Bei der Formulierung der Anforderungen für alle Use Cases wird berücksichtigt, dass die Erkenntnisse des Projektes sowohl im Labormaßstab als auch im Probebetrieb des Versuchsträgers *advanced TrainLab* gewonnen werden.

Das Ziel dieses Use Cases besteht darin, die Funktionen der JRU durch eine blockchainbasierte Software abzubilden. Durch Einsatz dieser Software auf im Zug vorhandenen Rechnern mit freien Rechenkapazitäten kann die physische JRU eingespart werden.

3.4.2 Funktionale Anforderungen

Alle funktionalen und nicht-funktionalen Anforderungen an den JBR wurden in einer Systemspezifikation festgehalten. Grundlage für die funktionalen Anforderungen ist das UML Use-Case-Diagramm in Abbildung 3.12, welches die Interaktion verschiedener Parteien mit dem JBR zeigt.

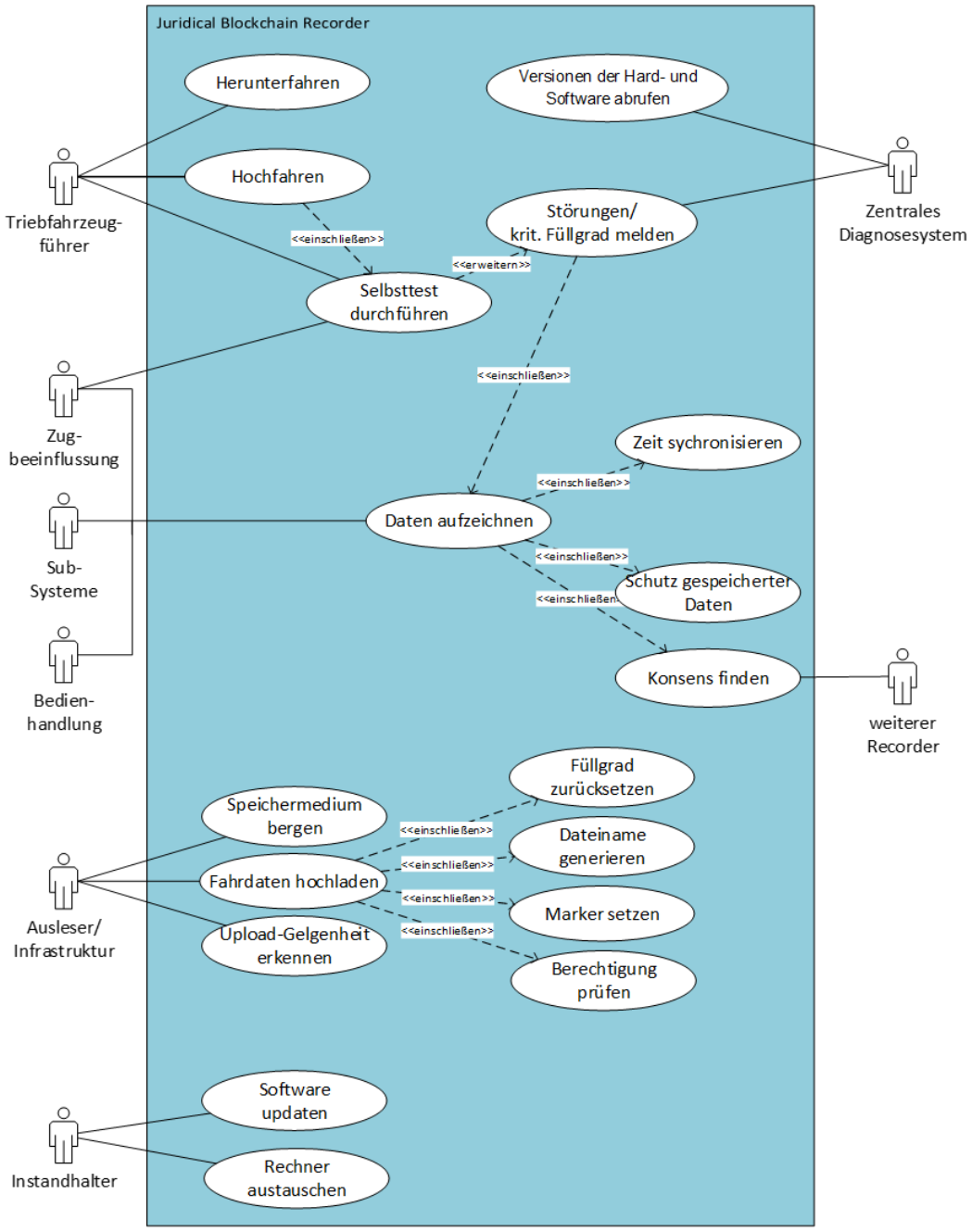

Abbildung 3.12: Use-Case-Diagramm zum JBR

Als Beispiel zeigt die folgende Tabelle das grundsätzliche Szenario „Daten aufzeichnen“.

Name	**Daten aufzeichnen**
Kurzbeschreibung	Der JBR registriert die relevanten Fahrdaten (siehe Unterabschnitt 3.3.1.3), die über verschiedene Schnittstellen (siehe Unterabschnitt 3.3.1.2) an ihn kommuniziert werden.
Akteure	Subsysteme im Fahrzeug -- insbesondere die Zugsteuerung, Fahrzeugbediener
Vorbedingungen	Der JBR besitzt eine Schnittstelle zu den Fahrdaten. Die Subsysteme senden die aufzuzeichnenden Fahrdaten an den JBR.
Fachlicher Auslöser	Aufzuzeichnende Nachrichten werden vom JBR als solche erkannt.
Ergebnis	Daten sind im Speichermedium des JBR gespeichert.
Hauptszenario	Auf dem Fahrzeugbus gesendete Nachrichten werden vom JBR erkannt. Dabei werden sie ihrem Inhalt nach klassifiziert und in entsprechenden Speicherbereichen abgelegt.
Alternativszenarien	Aufzuzeichnende Informationen, die über andere Schnittstellen als dem Fahrzeugbus mit dem JBR verbunden sind, werden erkannt und gespeichert.
Nachbedingungen	Der JBR ist rechtzeitig bereit um weitere Nachrichten zu erkennen und abzuspeichern.

Als weitere Use Cases wurden „Datenentnahme“, „Konsens finden“, „Aktualisieren und Rechner austauschen“, „Selbstest durchführen“, „Störungen melden“ sowie „Hoch-/Herunterfahren“ beschrieben. Aus den Use Cases wurden dann detaillierte Anforderungen mit eindeutigen Identifikatoren abgeleitet.

Identifikator	`#REQ-RC_UC3_SchnittstellenFahrdaten_#DEF0003:01`
Titel	**Datenquelle ETCS**
Beschreibung	Die Aufzeichnungsdaten müssen dem JBR vom ETCS-Fahrzeugsystem über ein Datenübertragungssystem übertragen werden.
Ursprung	DB_LH_ETCS Teil-LH 4.2.27
Querbezüge	—
Hinweise	Die Konformität der Aufzeichnungsdaten zu Subset 27 ist zu gewährleisten.

Nicht-funktionale Anforderungen an Verlässlichkeit (Zuverlässigkeit, Verfügbarkeit, Wartbarkeit, *Safety* und *Security*), die Beständigkeit des Speichermediums, Brand- und Umweltschutz, On-Board-Anwendung und Zulassung wurden ebenfalls detailliert dargelegt. Die folgende Tabelle zeigt ein Beispiel zur Verfügbarkeit.

Identifikator	`#REQ-RC_UC3_Verfügbarkeit_#DEF0006:01`
Titel	**Verfügbarkeit**
Beschreibung	Die Verfügbarkeit beträgt $\geq 99,99$ %
Ursprung	DB_LH_JRU Kapitel 4.3 FAS-Bo.4.3.5
Querbezüge	—
Hinweise	—

3.4.3 Vorhersagbarkeit

Die Echtzeitfähigkeit (< 1 Sekunde Block-Zyklus-Zeit) ist eine der Kernanforderungen für einen *Juridical Recorder*. Mit den üblichen auf DLT basierenden Anwendungen, vgl. zum Beispiel *Bitcoin*[1] oder *Ethereum*[2], sind solche Anforderungen nicht realisierbar, da deren Zeit für die Konsensfindung im Bereich von mehreren Minuten bzw. vielen Sekunden liegt und auch architekturbedingt nicht wesentlich beschleunigt werden kann. Hinzu kommt, dass bei normalen auf DLT basierenden Anwendungen die Finalität nur stochastisch, aber nicht deterministisch garantiert werden kann. Dies bedeutet praktisch, dass es Teilnehmer geben könnte, die auch nach einer längeren Zeit vom allgemeinen Konsens des Netzwerks abweichen könnten, wenn auch nur mit einer geringen Wahrscheinlichkeit. Für eine JBR-Anwendung sollte die Finalität aber deterministisch sein. Dies macht die Entwicklung eines spezifischen JBR-Protokolls unumgänglich.

Für die Umsetzung des JBR wurde daher ein Einigungsprotokoll entwickelt, welches auf die Tolerierung von beliebigen, auch als byzantinisch bezeichneten, Fehlern ausgelegt ist. Dies ermöglicht es JBR-Replikate kolloziert mit anderer für den Betrieb relevanter Software auf bereits vorhandenen Systemen zu betreiben. Selbst wenn einzelne Replikate durch einen Software- oder Hardwarefehler sich nicht mehr protokollkonform verhalten, werden weiterhin Nachrichten zuverlässig durch den Verbund des JBR aufgezeichnet. Weiterhin wird lediglich ein schwach echtzeitfähiges Kommunikationssystem benötigt. Im Falle des entwickelten Prototyps wurde hierzu auf Ethernet und TCP/IP zurückgegriffen. Dies ermöglicht es beispielsweise, wie im Testaufbau nachgewiesen, Nachrichten eines MVB zu protokollieren, ohne eine Änderung an dessen Kommunikationsplan vorzunehmen. Auch eine Einspeisung von Nachrichten anderer Kommunikationssysteme wird unterstützt.

[1] `https://bitcoin.org/`, besucht am 23.12.2022
[2] `https://ethereum.org/`, besucht am 23.12.2022

3.4.4 Anforderungen zu Sicherheit und Zuverlässigkeit

Zur Betrachtung der Sicherheit und Zuverlässigkeit wurden verschiedene Szenarien betrachtet, in denen die Leistung des JBR mit der einer JRU verglichen wurde. Grundsätzlich werden dabei zwei Szenarien unterschieden:

1. Die JRU fällt aus, d. h. die Daten werden nicht oder nicht mehr vollständig aufgezeichnet. Ein Unfall passiert, bevor die JRU instandgesetzt wird.
2. Ein Unfall passiert, die JRU wird so geschädigt, dass die Daten nicht oder nicht mehr vollständig zur Verfügung stehen.

Weitere Szenarien wurden zunächst nicht im Detail betrachtet, da sie deutlich unwahrscheinlicher sind. Nach dem Lastenheft der DB wird für eine JRU eine mittlere Zeit zwischen Ausfällen (MTBF) von 180.000 h gefordert bei einer mittleren Fehlerbehebungszeit (MTTR) von 18 h, dies entspricht einer Verfügbarkeit A von ca. 0,9999 (vgl. `#REQ-RC_UC3_Verfügbarkeit_#DEF0006:01`).

Dabei wird angenommen, dass die o. a. MTTR die reine Austauschzeit auf dem Fahrzeug bezeichnet, denn die JRU auf dem Zug wird nicht sofort repariert, d. h. der Zug setzt seine Fahrt trotz defekter JRU fort. Die wahre Zeit bis zum Beheben des Fehlers (seit Diagnose) ist wesentlich länger, z. B. 18 h (bei ununterbrochenem Betrieb). Nimmt man an, dass der Zug oder die Lok durchschnittlich 16 h pro Tag im Einsatz ist, dürfte die wahre MTTR in der Größenordnung von einem Tag liegen, d. h. MTTR=24 h. Wir verwenden hier eine MTTR von 18 h, um mit den ursprünglichen Anforderungen konform zu bleiben.

Im Fall eines Unfalls ist die JRU aufgrund ihrer Spezial-Hardware z. B. besonders vor Schock und Brand geschützt, z. B. gegen Feuer bis zu 700 °C über eine Dauer von 5 Minuten. Der Einfachheit halber wird angenommen, dass der Schutz der JRU perfekt ist, der der JBR aber nicht.

Die maßgebliche Ausfallart ist, dass die Daten der JRU nach einem meldepflichtigen Ereignis, in der Regel ein Unfall, nicht zur Verfügung stehen oder verfälscht sind.

Entscheidend ist die Suche nach gemeinsamen Ursachen, die alle Computer der JBR betreffen können. Hierbei ist zu beachten:

- Umweltbedingungen, die alle Computer, auf denen der JBR implementiert ist, nachteilig beeinflussen können. Diese werden jedoch in der Regel durch die Einhaltung der Anforderungen der EN 50125 [12] abgedeckt. Diese Norm ist verbindlich für alle Komponenten von Schienenfahrzeugen.
- Einflüsse durch elektromagnetische Strahlung. Durch die EN 50121 [11] werden Anforderungen hinsichtlich Emission und Strahlungsfestigkeit gestellt. In der Regel werden damit gemeinsame Einflüsse abgedeckt. Alle Komponenten von Schienenfahrzeugen müssen wiederum der EN 50121 [11] genügen, wodurch elektromagnetische Beeinflussungen nicht weiter untersucht werden müssen.

- Mechanische Einflüsse z. B. beim Aufprall als Folge eines Unfalls
- Feuer, z. B. infolge eines Brands nach einem Unfall

Die Analyse eines einfachen Markov-Modells ergab, dass die verteilte Struktur des JBR schon bei zwei Knoten und ohne spezielle Hardware dazu führt, dass die Zuverlässigkeit mindestens um einen Faktor 100 besser ist als bei der JRU. Da das Ziel ist vier verteilte Knoten im Anwendungsfall zu nutzen, ist die Zuverlässigkeit des JBR also weit besser als die der JRU.

Für das zweite Szenario wurde eine statistische Abschätzung über die schwere und Frequenz von Unfällen getätigt, die einen Ausfall des JBR nach sich ziehen könnten. Damit die JBR-Lösung mindestens genauso gut ist wie die JRU, muss diese Wahrscheinlichkeit in der Größenordnung 0,0001 liegen, d. h. in ein oder zwei Unfällen aus 10.000 Unfällen werden alle JBR-Knoten zerstört. Eine Analyse der durch die UIC gemeldeten Unfälle ergab die folgende relative Häufigkeit für Unfälle, bei denen mehrere Knoten zerstört werden können, unter der Annahme, dass die Schwere des Unfalls nur dann die erforderliche Hardware zerstört, wenn auch Todesfälle auftreten.

Hierzu ist anzumerken:

- Beim Aufprallen und Aufeinanderprallen von Schienenfahrzeugen können durchaus zwei Knoten zerstört werden. In der Regel ist die Mitte der Lokomotive oder die Mitte des Triebzuges ein vor Aufprall geschützter Ort. Man müsste hier die Knoten mit entsprechendem Abstand installieren. Bei Triebzügen ist dies möglich, sodass man ohne weitere Analyse davon ausgehen kann, dass bei Schäden durch Aufprall nicht alle vier Knoten zerstört werden. Inwiefern bei einzelnen Lokomotiven durch starkes Aufprallen nicht doch alle Knoten beschädigt werden können, lässt sich schwer einschätzen.
- Bei Feuer wurde kein Beitrag berechnet, da in der Stichprobe keine Toten durch Feuer berichtet wurden. Im Fall von Triebzügen wird mit einer Wahrscheinlichkeit nahe Eins immer einer der vier Knoten auslesbar bleiben. Zudem ist festzustellen, dass in modernen Lokomotiven – insbesondere denen, die mit ETCS ausgerüstet sind – Brandverhütungs- und teilweise Brandbekämpfungsmaßnahmen implementiert sind.

Unfallszenario	**Anteil des Szenarios an allen Unfällen**	**Anteil von Unfällen mit Toten**	**Anteil des Szenarios mit tödlichem Ausgang von allen Unfällen**
Entgleisung: Anteil 2,5 %	2,50 %	28,7 %	0,72 %
Zusammenstoß mit anderem Zug	0,50 %	63,6 %	0,32 %
Feuer	0,60 %	0,00 %	0,00 %
Summe	–	–	1,04 %

Tabelle 3.85: Anteile von Unfallszenarien

Unter der Annahme der Verteilung von Knoten an verschiedenen Enden des Zuges ergibt sich dann eine Wahrscheinlichkeit von $1\ \% \cdot 1\ \% = 10^{-4}$ für den gleichzeitigen Ausfall aller JBR-Knoten, was den Anforderungen genügt.

Allerdings gibt es auch noch weitere Faktoren, die die Verfügbarkeit der JBR-Lösung noch verbessern würden, allerdings hier numerisch nicht mit betrachtet wurden: Z. B. kann man bei der JBR-Lösung die Daten zusätzlich an der Streckenseite spiegeln, z. B. immer, wenn eine Breitbandverbindung zur Verfügung steht. Oder man kann die aktuellen Daten immer an die Streckenseite übertragen, wenn die JBR eine außergewöhnliche Betriebssituation erkennt z. B. eine Notbremsung.

Das zusammenfassende Fazit der Betrachtung von Use Case 3 ist, dass alle eingangs erwähnten Vorhaben erfüllbar sind, sofern die technische Umsetzung den erarbeiteten Anforderungen genügt.

4 Softwaretechnische Basis

Dieses Kapitel stellt die Identifikation und Auswahl der Basistechnologien vor. Die Ergebnisse dieser Arbeiten wurden bereits als wissenschaftliche Veröffentlichung auf einer internationalen Konferenz publiziert [50].

4.1 Identifikation von Basistechnologien

Für alle Use Cases müssen die grundlegenden bzw. zu entwickelnden Technologien analysiert bzw. bestimmt werden. Zunächst werden Hintergründe zu Use Case 1 diskutiert, insbesondere die Realisierung und Infrastrukturen dezentraler digitaler Identitäten. Ferner werden die Grundlagen von traditionellen *Juridical Recording Unit*s (JRUs) sowie von gebräuchlichen Zugbussen untersucht. In Anbetracht der Zielsetzung von Use Case 2 und Use Case 3, spielen außerdem *Distributed-Ledger*-Technologien (DLTs) eine vordergründige Rolle, weshalb die Grundlagen von Blockchains und Konsensprotokollen erörtert werden.

4.1.1 *Asset Identity* (Use Case 1)

Eine digitale und eindeutige Repräsentation eines Assets ist zum einen durch eindeutige Merkmale (z. B. eine Seriennummer einer Baureihe) und zum anderen durch Zusatzinformationen, die die Herkunft und die Validität dieser Daten nachprüfbar machen. Sehr verbreitet sind heutzutage die Zertifikate nach dem Standard X.509. Es gibt im Umgang mit den X.509-Zertifikaten allerdings erhebliche Nachteile, die eine offene Interoperabilität verhindern. Wie in der Einleitung zu Use Case 1 bereits beschrieben ist die Entwicklung von *Decentralized Identifier* [27] und *Verifiable Credential* eine sehr interessante Basis zur Umsetzung der gewünschten Funktionen des Use Case 1:

Diese Technologie erfordert ein dezentrales Register, das öffentliche Schlüssel und verifizierbare Informationen durch standardisierte Schnittstellennach dem DID-Standard bereitstellt. Das *IDunion*-Projekt ist als Kooperationspartner für das *RailChain* Projekt gewonnen worden, um mit Hilfe des *IDunion*-Netzwerks die Funktionen und die Interoperabilität des Use Case 1 zu demonstrieren (Abbildung 4.2).

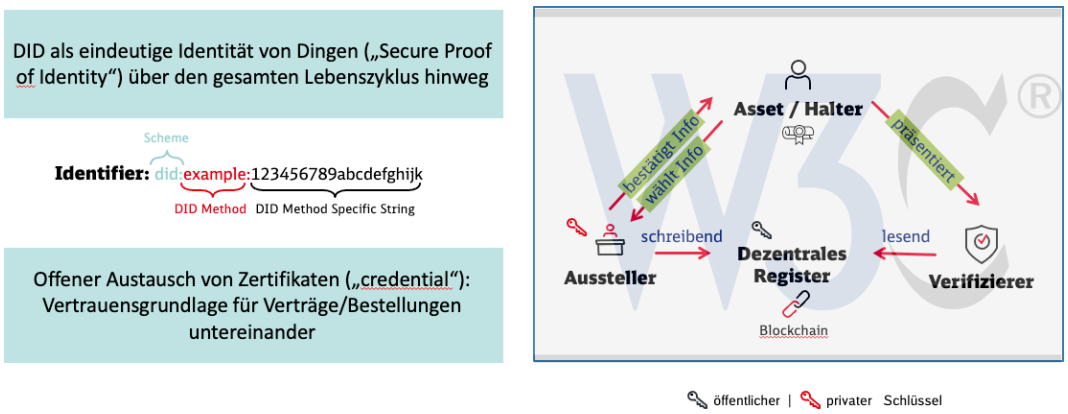

Abbildung 4.1: Merkmale und Prozesse von DIDs

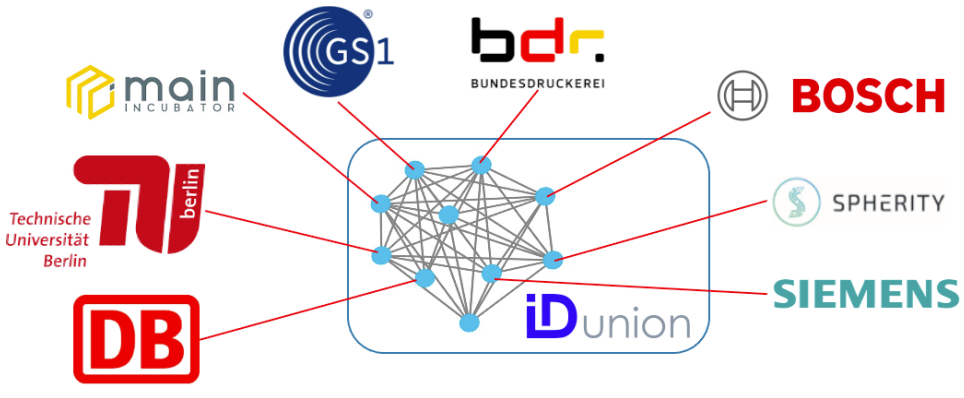

Abbildung 4.2: Netzwerk von *IDunion* für DIDs, realisiert mittels DLT

4.1.2 Juridical Recording Unit (Use Case 2 & Use Case 3)

Um zu verstehen, welche Funktionalität in Use Cases 2 und 3 durch einen blockchainbasierten Ansatz ermöglicht werden soll, muss zunächst die traditionelle JRU näher betrachtet werden. Die JRU ist die zentrale Einheit, die für das Aufzeichnen juristisch relevanter Informationen verantwortlich ist. Diese Informationen sind auftretende Ereignisse (engl. *Events)*, operative Parameter, sowie deren Zeitstempel. Anhand dieser Daten kann im Falle eines meldepflichtigen Ereignisses wie einem Unfall der genaue Ablauf rekonstruiert werden. Es gibt keine globale, legal bindende Definition der JRU, stattdessen erstellen die nationalen Eisenbahnunternehmen ihre eigene Definition anhand von existierenden Standards wie etwa IEC 62625-1 [35].

Die Daten werden in einem Ringpuffer von fester Größe gespeichert, der im Flash-Speicher gehalten wird. Diese Form der Speicherung hat zur Folge, dass die ältesten Daten im Ringpuffer bei Speichermangel überschrieben werden. Da im Allgemeinen die neusten Daten von größter Bedeutung sind, ist dies jedoch nicht problematisch. Autorisiertes Personal kann mittels eines speziellen physischen Schlüssels auf die entsprechenden Schnittstellen der JRU zugreifen und die Daten extrahieren. Dieses Auslesen der Daten kann bei Bedarf zu Unterstützung der Wartung oder zur Unfallanalyse durchgeführt werden.

Die JRU befindet sich an einem zentralen Punkt im Zug, wie etwa der Lokomotive und ist an den zentralen Datenbus angeschlossen. Über diesen Bus werden die juristisch relevanten bzw. sicherheitsrelevanten Signale übertragen. Welche Signale konkret von der JRU aufgezeichnet werden sollen, wird während der Installation bzw. der Wartung über eine Konfigurationsdatei (*Node Supervisor Database* – NSDB) festgelegt. Beispiele für die aufzuzeichnenden Informationen sind etwa die Geschwindigkeit des Zuges, der Druck in der Hauptluftleitung, sicherheitsrelevante Befehle des Zugführers und Signale des Zugbeeinflussungssystems. Dieses empfängt Daten von Sensoren, führt Vorberechnungen durch und leitet die Ergebnisse über den Bus weiter, damit sie für den Zugführer angezeigt und von der JRU aufgezeichnet werden können.

Der Zug fährt auch bei Beschädigung oder Ausfall der JRU normal weiter. Die JRU ist hauptsächlich abgesichert gegen unautorisierten Zugriff, physischen Schaden und Manipulation durch Integritätsmaßnahmen wie Prüfsummen. Die empfangenen Signale werden vor dem Abspeichern nicht verifiziert oder kontrolliert. Empfangene Daten werden auf Doppelungen gefiltert, sodass z. B. die Geschwindigkeit nur aufgezeichnet wird, wenn sich der Wert ändert. Datenverlust und -manipulation, z. B. während oder nach der Extraktion, sind möglich.

Ein Ziel des Projektes besteht darin, durch einen verteilten Aufbau des Systems im Zug eine vergleichbare Zuverlässigkeit für den Erhalt der Daten zu gewährleisten. Dieses Ziel ist auch aus Kostengründen interessant, da das Prinzip von einer geringeren Zuverlässigkeit einzelner Computerknoten ausgehen kann, die aus Standardhardware bestehen können. Der Nachweis der vergleichbaren Zuverlässigkeit ist in der Eisenbahnfachzeitschrift *Signal und Draht* veröffentlicht worden und wird an dieser Stelle zitiert [3].

4.1.3 Kommunikationsbus

Das Zugkommunikationsnetz wird genutzt, um verlässlich Informationen im Zug zu verteilen. Bussysteme, die in modernen Zügen zum Einsatz kommen, sind *ProfiBus*, *ProfiNet* und CAN-Busse, am weitesten verbreitet ist allerdings der *Multifunction Vehicle Bus* (MVB), der Geräte innerhalb eines Wagens verbindet, in Kombination mit dem *Wire Train Bus* (WTB), der die jeweiligen Wagen verbindet.

Der MVB wird von Siemens und ABB eingesetzt und ist in IEC 61375-3-1 [34] standardisiert. Er ist ein synchronisierter Datenbus mit einer Anführer/Folger-Kommunikation (*Master* und *Slave*). Der MVB-Master gibt den Buszyklus vor und fragt die verbundenen Geräte nach neuen Daten an. Der MVB transferiert Prozess-, Nachrichten- und Supervisory-Daten. Für die JRU sind die periodisch übertragenen Prozessdaten relevant. Für eine weitergehende Beschreibung des MVB siehe Unterabschnitt 3.1.1.

4.1.4 Blockchain und Einigungsprotokolle

Viele der Eigenschaften von vollständigen DLT-Plattformen sind für die Use Cases 2 und 3 nicht relevant. Wir interessieren uns hauptsächlich auf die unterliegende Datenstruktur zur Speicherung der Informationen sowie die Zugriffs- und Mitgliederverwaltung. Es gibt verschiedene Möglichkeiten, die Datenstruktur zu gestalten. Wir gehen im Folgenden auf die klassische Blockchain näher ein, die unsere Anforderungen aus den Definitionen der Use Cases gut erfüllt. Allgemein formuliert ist eine Blockchain eine replizierte, verteilte Datenstruktur, die Daten in Form von Transaktionen speichert, die in einem *Peer-to-Peer*-Netzwerk erstellt werden. Diese Transaktionen werden dann in Blöcke zusammengefasst, die mit einem Zeitstempel versehen werden. Jeder Block enthält mehrere Transaktionen und wird durch seinen kryptographischen Hashwert identifiziert. Ein Block enthält außerdem immer eine Referenz auf den vorherigen Block, indem dieser Hashwert dort abgespeichert wird. Dies ergibt eine lineare, chronologische Kette an Blöcken, nämlich die Blockchain.

Die Erstellung des nächsten Blocks wird durch sogenannte Einigungsprotokolle durchgeführt. Diese sichern die Integrität und die Konsistenz der Daten und somit der gesamten Blockchain, um eine unveränderbare Aufzeichnung der Transaktionen zu ermöglichen. Die bekanntesten Kategorien von Einigungsprotokollen sind *Proof of Work*, *Proof of Stake* und Einigungsprotokolle mit BFT (Byzantinische Fehlertoleranz).

Blockchains bzw. DLT-Plattformen können anhand ihrer Zugriffsberechtigungen in *permissionless* oder *permissioned* eingeteilt werden. In *permissionless* Blockchains können neue Benutzer ohne Überprüfung ihrer Identität dem Netzwerk beitreten, auf die Daten der Blockchain zugreifen und neue Daten/Transaktionen schreiben. Diese Form wird häufig für Kryptowährungen verwendet. Die bekanntesten Beispiele sind *Bitcoin* mit dem Einigungsprotokoll *Proof of Work* und *Ethereum* mit *Proof of Stake*. In einer *permissioned* Blockchain sind hingegen alle Teilnehmer bekannt und authentifiziert. Diese Form wird häufig im Industriebereich verwendet, etwa

für die Verwaltung von Lieferketten, und ein bekanntes Beispiel ist *Hyperledger Fabric*, welches den Einsatz von BFT-Protokollen unterstützt.

In byzantinisch fehlertoleranten Einigungsprotokollen haben wir eine Menge N, $|N| = n$, Knoten, die einen verteilten Dienst replizieren. Um den Zustand des Systems konsistent zu halten, wird eine totale Ordnung über die Anfragen an diesen Dienst hergestellt. Eine bestimmte Anzahl f, $n = 3f + 1$ im Standardmodell, der Knoten im System können sich dabei beliebig fehlerhaft verhalten, ohne die Korrektheit des Gesamtsystems zu beeinflussen. BFT-Protokolle kommen oft im Blockchain-Kontext vor: je nach Einsatzgebiet sowohl die traditionellen Protokolle, die auf eine kleine, bekannte Anzahl an Teilnehmern ausgelegt sind, sowie skalierbare Protokolle, die eine räumlich verteilte Teilnehmermenge annehmen.

Das bekannteste BFT-Protokoll ist das etablierte Protokoll PBFT (*Practical Byzantine Fault Tolerance*) [6]. Hier wird aus der Menge der Teilnehmer einer als Anführer festgelegt, während die anderen Replikate sogenannte Backups sind. Clients sind authentifiziert und können Anfragen an den replizierten Dienst stellen. Die Replikate durchlaufen drei Phasen, um eine totale Ordnung über diese Anfragen zu erstellen: *Preprepare*, *Prepare* und *Commit*. In der *Preprepare*-Phase sendet der Anführer die ausgewählte Anfrage an die Backups via Broadcast, hierbei wird dieser Anfrage eine Sequenznummer zugeordnet. Die Backups senden dann eine *Prepare*-Nachricht über Multicast, um den Empfang dieser Anfrage mit dieser Sequenznummer zu bestätigen. Sobald ein Replikat mindestens $2f$ dieser *Prepare*-Nachrichten erhalten hat, sendet es eine *Commit*-Nachricht via Multicast, um diese Anfrage final zu akzeptieren. Nach dem Empfang von $2f + 1$ *Commit*-Nachrichten (einschließlich seiner eigenen) führt ein Replikat diese Anfrage aus. Das Ergebnis der Operation wird an den Client zurückgegeben, der auf mindestens $f + 1$ identische Antworten wartet, um Korrektheit sicherzustellen. Jedes der Replikate kann fehlerhaft oder unverfügbar werden, solange die Gesamtanzahl der Fehler kleiner oder gleich f bleibt. Dies schließt den Anführer mit ein, was auf den Backups erkannt werden kann, z. B. durch Zeitbeschränkungen (engl. *timeouts*), wenn der Anführer keine Anfragen mehr weiterleitet. In diesem Fall wird über das *View-Change*-Subprotokoll ein neuer Anführer bestimmt, um den normalen Betrieb wiederherzustellen. Auch hierfür muss eine Mehrheit von insgesamt $2f$ Replikaten zustimmen, um den neuen Anführer zu wählen. PBFT bietet Korrektheit im asynchronen Kommunikationsmodell und garantiert Fortschritt (*Liveness*) im partiell synchronen Modell.

4.2 Analyse von Fehlermodell und DLT

Um die in Kapitel 3 beschriebenen Use Cases erfolgreich umzusetzen, muss zuerst das Systemmodell klar definiert werden. Dies kann sich zwischen den jeweiligen Use Cases und auch zwischen den jeweiligen Prototypen je nach Fokus leicht unterscheiden. Hierzu gehört insbesondere auch das Fehlermodell, also welches Verhalten von den einzelnen Komponenten erwartet wird und was für Fehlverhalten toleriert werden muss. Darüber hinaus müssen die Anforderungen an die verwendete *Distributed-Ledger*-Technologie (DLT) erfasst werden, existierende Implementierungen auf Eignung überprüft und gegebenenfalls Eigenschaften für eine angepasste Implementierung ergründet werden.

Das Aufzeichnen von juristisch relevanten Daten durch die JRU beinhaltet zwar Daten, die als sicherheitskritisch angesehen werden können (z. B. die Geschwindigkeit des Zuges oder das Auslösen der Notbremsen), ist aber selber keine sicherheitskritische Funktionalität. Bei der traditionellen JRU liegt der Fokus im Allgemeinen auf dem physischen Absichern des Gerätes im Zug. So ist das Gerät etwa robust konstruiert, um z. B. gegen Feuer oder Wuchtschaden geschützt zu sein, und der Zugriff auf die Daten wird auf berechtigte Mitarbeiter beschränkt, die mit einem entsprechenden Schlüssel ausgestattet sind, um auf den USB-Anschluss zuzugreifen. Die Integrität der Daten wird durch Prüfsummen sichergestellt, die über die empfangenen und aufgezeichneten Daten berechnet wurden. Ein Ausfall der JRU während einer Fahrt wird nicht als kritisch angesehen und Züge fahren bei Störungen oder Ausfällen der JRU unverändert weiter, ohne dass (korrekte) Daten aufgezeichnet werden können.

Braband et al. [3] haben analysiert, wie die Verfügbarkeit (engl. *Availability*) eines *Juridical Recorder* beeinflusst wird, wenn die Funktionalität auf mehrere Knoten verteilt wird. Basierend auf den Werten der JRU für die mittlere Zeit zwischen Ausfällen und der mittleren Fehlerbehebungszeit haben sie die Verfügbarkeit der JRU ermittelt, die ein verteilter *Juridical Blockchain Recorder* (JBR) ebenfalls erreichen sollte. Sie haben dafür den Sicherheitsschutz (*Safety*) der JRU als perfekt angenommen, während für den JBR, der auf normalen, fahrzeugtauglichen Rechnern laufen soll, normale Sicherheitsannahmen gelten. Die insgesamt schwachen Sicherheitsmechanismen (*Security*) der JRU werden nicht weiter betrachtet und der blockchainbasierte Manipulationsschutz des JBR wird in jedem Fall als besser angenommen.

Die Analyse betrachtet verschiedene Fehlerszenarien von „meldepflichtigen Ereignissen“ wie etwa Unfällen, die zu einer Beeinträchtigung der Datenvollständigkeit und -unversehrtheit führen können. Umweltbedingungen, die alle Computer des JBR beeinflussen, oder der Einfluss von elektromagnetischer Strahlung sind durch die Standards EN 50125 [12] bzw. EN 50121 [11] abgedeckt und werden nicht betrachtet. Relevant sind etwa mechanische Einflüsse oder Feuer. Durch mathematische Modellierung konnte ermittelt werden, dass bei der Ausführung des JBR auf mehreren (N=4), über den Zug verteilten Rechnern eine nicht niedrigere Verfügbarkeit erreicht werden kann als für die JRU. Die Wahrscheinlichkeit, dass bei einem Unfall mehrere oder gar alle JBR-Rechner beschädigt werden, ist

ausreichend gering. Des Weiteren besteht die Möglichkeit, die aufgezeichneten Daten bereits während der Fahrt zu exportieren, sodass selbst bei Datenverlust oder -beschädigung auf dem Zug diese Daten weiterhin zur Verfügung stehen. Die Analyse zeigt somit, dass eine solche Digitalisierung der JRU in Form eines blockchainbasierten, verteilten JBR eine angemessene Verfügbarkeit bietet.

Man unterscheidet im Allgemeinen zwischen CFT (*Crash Fault Tolerance*, Fehlertoleranz gegenüber Abstürzen) und BFT (Byzantinische Fehlertoleranz). Während im ersten Modell ausgefallene Komponenten sich nicht weiter negativ auf die Ausführung auswirken, umfasst byzantinisches Fehlverhalten beliebiges Verhalten von betroffenen Komponenten. Dies kann beispielsweise durch einen Hardware- oder Softwarefehler ausgelöst werden, aber auch durch einen Angreifer, der sich unbefugten Zugriff zum System beschafft hat. Im Zug, wo wir es mit einem geschlossenen System zu tun haben und im Normalfall keine Komponenten nach Belieben hinzugefügt werden, ist absichtlich bösartiges Verhalten weniger im Fokus als eventuelle Hardwarefehler oder gegenseitige Beeinflussungen, die auf den geteilten Geräten entstehen können. Für Verfügbarkeit und Zuverlässigkeit müssen wir dieses beliebige Fehlverhalten tolerieren können und können uns nicht nur auf Ausfälle konzentrieren.

Für die Use Cases 2 und 3 verteilen wir die Funktionalität des blockchainbasierten Datenschreibers auf mehrere fahrzeugtaugliche Rechner, wie in der Analyse von Use Case 3 angenommen. Diese sind an den Kommunikationsbus, über den die aufzuzeichnenden Daten gesendet werden, angeschlossen. Außerdem sind sie mit einer separaten Ethernet-Verbindung angebunden. Über diese Ethernet-Verbindung findet die Kommunikation für das Einigungsprotokoll zur Erstellung der Blockchain statt, um zu vermeiden, dass diese Nachrichten mit einer potentiell großen Menge an Daten bzw. individuellen Nachrichten die Buskommunikation überfüllen und dort zu Datenverlust führen. Wir treffen somit die Annahme, dass diese Kommunikation zwischen den Knoten partiell synchron ist. Das bedeutet, dass die Verbindungen asynchron sind und der Empfang von Nachrichten um eine konstante, aber unbekannte Zeit verzögert sein kann, nach der das System temporär synchron ist.

Die aufzuzeichnenden Daten werden über den Kommunikationsbus empfangen, typischerweise der MVB. Dieser Bus ist zeitgesteuert und die Kommunikation ist somit synchron. Alle Sicherheitsanforderungen der Buskommunikation werden durch die Erfüllung der entsprechenden Standards gewährleistet. Es findet keine Authentifizierung der Datenquellen statt und es kann nicht festgestellt werden, welche Komponente ein Signal gesendet hat. Es ist möglich, dass Fehler wie z. B. sog. *Bitflips* in der Kommunikation auftreten. Wir fassen alle Signale, die während eines Buszyklus empfangen werden, zu einem einzelnen *Request* zusammen. Im Normalfall lesen alle Knoten die gleichen Daten vom Bus und erhalten somit den gleichen *Request*. Die über den Bus gesendeten Daten sind eindeutig, z. B. durch Zeitstempel, aber Duplikate im System können dadurch entstehen, dass mehrere Replikate Eingaben identisch lesen und weiterleiten. Hier gilt, dass wir die duplizierten Eingaben filtern wollen, die wir durch den identischen Empfang auf mehreren Knoten erhalten, um die Gesamtlast auf dem System zu verringern. Gleichzeitig müssen

wir sicherstellen, dass alle Daten, die auf den Knoten empfangen werden, sicher und persistent gespeichert werden.

Durch die Struktur der Blockchain, in der die Daten gespeichert werden, wird nachträgliche Veränderung oder nachträgliches Einfügen von Daten verhindert. Die Reihenfolge der aufgezeichneten Daten, d. h. in der Blockchain, ist nicht garantiert die gleiche, in der die Daten über den Bus empfangen wurden. Diese Ordnung wird bei Bedarf nach dem Export der Daten in Laboranalysen über den aufgezeichneten Zeitstempel hergestellt.

Alle Knoten haben ein asymmetrisches Schlüsselpaar für kryptographische Operationen und alle Nachrichten, die von den Knoten ausgetauscht werden, sind asymmetrisch signiert um ihre Integrität sicherzustellen. Wir gehen davon aus, dass diese kryptographischen Verfahren nicht von einem Angreifer in angemessener Zeit gebrochen werden können und somit sicher sind.

Für die Fehlerannahmen betrachten wir das Verhalten der Knoten im verteilten Datenschreiber und dem restlichen System, insbesondere des Kommunikationsbusses und dessen Komponenten wie dem MVB-Master, separat. Für die Knoten gilt das byzantinische Fehlermodell: Wir benötigen eine Menge N von $|N| = n \geq 3f + 1$ Knoten, von denen sich zu jedem Zeitpunkt maximal f gleichzeitig fehlerhaft (byzantinisch) verhalten dürfen. Dieses Verhalten umfasst das Verzögern, Duplizieren, Unterschlagen, Umsortieren und Korrumpieren von Nachrichten durch fehlerhafte Knoten oder Angreifer mit dem Ziel, das System in einen inkonsistenten Zustand zu versetzen. Außerdem können fehlerhafte Knoten, die Daten vom Bus lesen, diese unterschlagen, duplizieren, umsortieren oder sogar neue Daten erstellen. Dies kann beispielsweise bedeuten, dass ein Knoten während eines Buszyklus keine Daten empfängt oder dass Daten aus einem Zyklus während eines anderen Zyklus gelesen werden. Außerdem können Knoten Daten lesen, die in dieser Form nicht von anderen Knoten gelesen wurden. Das Erstellen von Daten, die nicht über den Bus gesendet worden sind, kann beispielsweise durch einen *Denial-of-Service*-Angriff (siehe auch Unterabschnitt 4.3.3) motiviert sein. Dabei kann entweder die im System provozierte Überlast zu dessen Ausfall führen, oder es kann die Analyse der JRU-Daten nach einem Unfall verfälscht werden. Das Filtern von Duplikaten soll nicht verhindern, dass solche einzeln empfangenen Daten aufgezeichnet werden, da diese Daten wichtige Informationen für die Laboranalyse enthalten können. Das Erkennen von solchen neu hinzugefügten Daten und das Erstellen bzw. Wiederherstellen der korrekten Reihenfolge von Signalen wird durch Laboranalysen nach dem Export der Daten durchgeführt und liegt nicht im Aufgabenbereich des JBR. Alle *Requests* sollten in Kombination mit einer Knoten-ID, der diese Daten empfangen hat, aufgezeichnet werden. Im Standardfall ist dies die ID des Anführers, der den *Request* weiterleitet. Es kann aber auch ein Backup-Knoten sein, der einen einzeln empfangenen *Request* weiterleitet. So kann auch nachträglich festgestellt werden, ob ein als fehlerhaft identifizierter Knoten potentiell korrumpierte *Requests* weitergeleitet hat.

Die für die Umsetzung der Use Cases verwendete DLT muss auf deren Bedürfnisse angepasst werden.

Wir haben eine begrenzte Anzahl an relevanten Komponenten im System, die an dem *Juridical Recorder* beteiligt sind. Alle Komponenten sind bekannt und werden im Optimalfall nicht bzw. erst nach langer Benutzungsdauer ausgetauscht. Da die aufzuzeichnenden Daten möglicherweise sicherheitskritisch oder vertraulich sind, sollten sie nur für berechtigte Teilnehmer verfügbar sein.

Wir haben bereits analysiert, dass das byzantinische Fehlermodell gut auf unser System passt. Einigungsprotokolle basierend auf *Proof of Work* oder *Proof of Stake* sind aufgrund ihrer hohen Latenz bzw. der Voraussetzung einer Kryptowährung nicht für den Einsatz im Eisenbahnkontext geeignet.

Diese beiden Aspekte passen gut auf eine private, *permissioned* DLT-Plattform. Hier können BFT-Protokolle als Grundlage für die Einigung verwendet werden. Diese Protokolle sind zwar weniger skalierbar als PoW oder PoS, aber dafür performanter, was wiederum positiv für die Erfüllung der Echtzeitanforderungen ist.

Wir haben die Anforderungen an die DLT-Plattform definiert und verschiedene bekannte DLT-Plattformen diesbezüglich verglichen. Hierfür haben wir *Ethereum*, *Hyperledger Fabric*[1] und *IOTA*[2] ausgewählt. Die Anforderungen sind:

- Verwendung einer effizienten, integritätswahrenden Datenstruktur;
- Eignung für den *permissioned* Einsatz;
- Verwendung von BFT-Protokollen zur Einigung möglich;
- effiziente, portierbare Serialisierung der Daten;
- keine Kosten für Transaktionen;
- kein notwendiger Einsatz von *Smart Contracts*, um möglichen Mehraufwand zu vermeiden;
- Modularität und Anpassbarkeit der Plattform, falls sich die Anforderungen der Use Cases im Laufe des Projektes ändern oder erweitern sollten.

Ethereum wurde 2013 gegründet und ist bekannt für seine Kryptowährung Ether. *Ethereum* kann sowohl *permissionless* als auch *permissioned* betrieben werden. Daten werden in der Struktur einer Blockchain gespeichert, aufgeteilt in *Tries*. *Smart Contracts* können in der Programmiersprache *Solidity* geschrieben werden, die entsprechenden Operationen müssen mit sogenanntem Gas bezahlt werden. Als Einigungsprotokoll kommt PoW zum Einsatz; ein Wechsel auf PoS ist geplant. Im *permissioned* Setting sind auch BFT-Protokolle möglich. Die Serialisierung geschieht über *Recursive Length Prefix* und durch Konfigurationen der Plattform ist es möglich, dass *Smart Contracts* nicht zwingend notwendig sind. Die Modularität und Anpassbarkeit von *Ethereum* sind eingeschränkt.

[1] `https://www.hyperledger.org/use/fabric`, besucht am 23.12.2022
[2] `https://www.iota.org/`, besucht am 23.12.2022

Hyperledger Fabric ist ein Framework zur Erstellung von *permissioned* Blockchains, das in verschiedenen industriellen Bereichen zum Einsatz kommt, wie z. B. in der Verwaltung von Lieferketten. Es ist explizit für einen modularen, erweiterbaren Aufbau angelegt. So ist das Einigungsprotokoll etwa austauschbar und es werden Protokolle mit CFT oder Protokolle mit BFT unterstützt. *Smart Contracts*, genannt *Chaincodes*, können in weit verbreiteten Programmiersprachen wie *Go* oder *Java* verfasst werden. Es fallen keine Kosten für Transaktionen oder die Ausführungen von *Smart Contracts* an. Die *Smart Contracts* sind allerdings für den allgemeinen Blockchain-Betrieb notwendig, etwa für die Verifizierung und Speicherung von Daten. Das Datenformat für Blöcke ist im portablen Format *Protocol Buffers* (auch *protobuf* genannt)[3] definiert.

IOTA ist eine DLT-Plattform, die auf das Einsatzgebiet des *Internet of Things* (IoT) abzielt. Hier sollen Geräte des IoT Transaktionen tätigen können, ohne dass der Benutzer selber aktiv werden muss. Auch *IOTA* hat eine Kryptowährung. Die unterliegende Datenstruktur ist der *Tangle*, ein direktionaler azyklischer Graph, der Referenzen mehrerer Transaktionen verknüpft. Transaktionen können erst aufgenommen bzw. bestätigt werden, wenn sie zwei zufällig ausgewählte vorherige Transaktionen bestätigen. So ist keine eindeutige Ordnung und Nachvollziehbarkeit garantiert. Außerdem ersetzt der *Tangle* das Einigungsprotokoll und es kann somit nicht für BFT eingesetzt werden. Es fallen keine Gebühren für Transaktionen an. Zum Zeitpunkt der Analyse benötigte *IOTA* einen zentralen Koordinator, der für die Erstellung von sogenannten *Milestones*, also signierten Transaktionen, verantwortlich ist und diese damit bestätigt. Außerdem gab es zu diesem Zeitpunkt keine Unterstützung für *Smart Contracts*.

Diese repräsentativen Plattformen haben alle gewisse Nachteile in ihrem Aufbau, sorgen teilweise für erheblichen Mehraufwand und sind zu unflexibel, um sie an das Einsatzgebiet im Eisenbahnkontext anpassen zu können. Wir setzen somit auf eine modulare, leichtgewichtige Eigenimplementierung, die die Grundlagen von Blockchains umsetzt. Sie verwaltet also eine Hashchain: eine durch Hashwerte verkettete Liste an Blöcken, in denen Daten abgespeichert werden können und die durch ein BFT-Einigungsprotokoll erstellt werden.

[3] `https://developers.google.com/protocol-buffers`, besucht am 23.12.2022

4.3 Spezifikation des Einigungsprotokolls

Wir stellen folgende Anforderungen an unser System und erläutern im Folgenden, wie diese erfüllt werden:

- R1 Ersetzen der JRU durch ein repliziertes System, das eine hohe Leistung trotz beliebiger Ausfälle bietet.
- R2 Ersetzen eines teuren, dedizierten Geräts durch die opportunistische Nutzung von Standard-Hardware im Zug, ohne Änderungen an der sicherheitskritischen Kommunikationsinfrastruktur.
- R3 Erfüllung der JRU-Anforderungen bei gleichzeitiger Vermeidung des Ausfalls von über den Bus empfangenen Daten sowie die rückwirkende Manipulation von aufgezeichneten Daten.
- R4 Sicherer und konsistenter Datenexport bei gleichzeitigem Schutz der Integrität und Speicherbereinigung (engl. *Garbage Collection*) ohne Verwerfen von Daten.

4.3.1 Ein Überblick über das ZugChain-System

Im Kontext unseres Systems „ZugChain“ [50] sind mehrere Parteien involviert: die Eisenbahnverkehrsunternehmen (EVU), Unternehmen die Züge oder Zugkomponenten herstellen und Bundesbehörden, die im Falle von Zwischenfällen die Verantwortung ermitteln. Keine einzelne Partei sollte die Möglichkeit haben, die aufgezeichneten Daten zu manipulieren. Da ein Zug ein heterogenes System ist, kommen die installierten Hardware- oder Softwarekomponenten nicht nur von einem Hersteller. Im Falle eines Vorfalls wird derzeit das bisherige Verhalten jeder sicherheitsrelevanten Komponente analysiert und anhand von JRU-Daten rekonstruiert. Wenn wir ein zertifiziertes, manipulationssicheres Gerät durch eine Software, die auf handelsüblicher Hardware läuft, ersetzen, bietet der neue Aufbau mehr Möglichkeiten für ungewollte und möglicherweise böswillige Eingriffe. Dementsprechend streben wir an, dass nicht ein einziges Unternehmen die Kontrolle über die Protokollierung und damit die Möglichkeit hat, andere für ein Fehlverhalten verantwortlich zu machen. Stattdessen soll allen Unternehmen eine Teilverantwortung eingeräumt werden.

Unser in Abbildung 4.3 dargestellter Ansatz ersetzt die JRU durch eine verteilte blockchainbasierte Lösung. Durch die Verwendung bereits installierter, fahrzeugseitiger Hardware als ZugChain-Knoten minimieren wir Änderungen am Zug. Diese Knoten, die über genügend Ressourcen verfügen, um das ZugChain-System auszuführen, können von jedem der Unternehmen stammen, was somit R2 erfüllt. Nachrichten werden als Signale über den Kommunikationsbus empfangen. Datenverluste auf dem Bus, z. B. das Löschen von über den Bus gesendeten Signalen oder das Blockieren des Datenempfangs auf Knoten, ist durch die Busspezifikationen abgedeckt und wird daher hier nicht weiter betrachtet. Nachrichten werden von

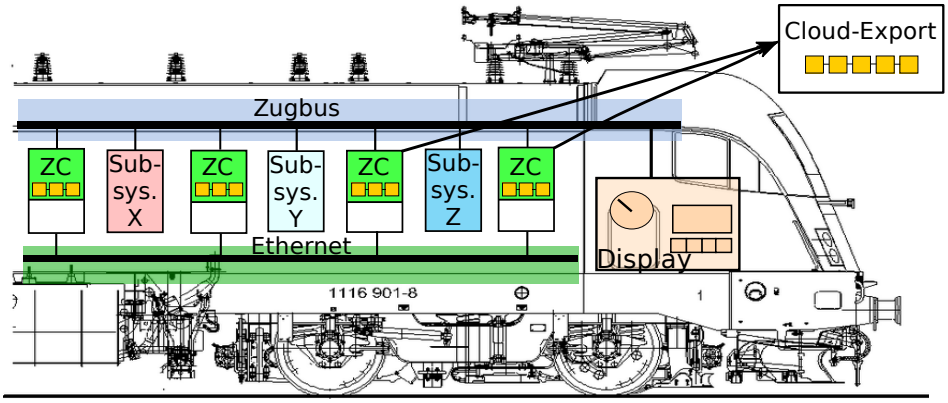

Abbildung 4.3: Verteilte, blockchainbasierte Lösung für eine JRU (jedes „ZC“ steht für einen ZugChain-Knoten)

allen Knoten unabhängig vom unzuverlässigen Bus gelesen. ZugChain zeichnet die gleichen Daten auf wie die JRU, d. h. wir verlangen keine Änderungen der Signale, nachdem sie z. B. von der Zugbeeinflussung gesendet wurden. Die Daten werden in der gleichen Busfrequenz und Qualität wie in der ursprünglichen JRU aufgezeichnet, sodass bestehende Anforderungen weiterhin erfüllt werden. Einige Daten werden von der JRU verschlüsselt empfangen und aufgezeichnet, was ZugChain identisch behandelt. Diese Daten können sensible Informationen enthalten, die vor der Busübertragung an der Datenquelle verschlüsselt und authentifiziert werden und nach dem Export nur für autorisierte Parteien verfügbar sind.

Die Daten werden über den Bus in einem Rohformat empfangen, aus dem wir die Signale ableiten. Dabei werden identische Transformationsschritte wie in der JRU verwendet, die auf Korrektheit überprüft wurden. Die Knoten empfangen, analysieren und filtern die Daten nach Relevanz und für eine höhere Effizienz, wie es bei JRUs üblich ist, z. B. um die Geschwindigkeit nur bei Änderungen zu protokollieren. Danach sind die vom Bus empfangenen Nachrichten eindeutig. Sobald die Nachricht transformiert wurde, wird sie an das BFT-Protokoll übermittelt. Durch die Verwendung einer Kombination aus einem BFT-Protokoll für die Einigung und einer Blockchain für die Integrität sichern wir sowohl gegen Fehler während des -Betriebs, z. B. gegen Unfälle, bei denen alle Blockchain-Knoten bis auf einen zerstört werden, als auch gegen Fehler bei der Einigung, die von der willkürlichen Beeinflussung durch beliebige Prozesse auf gemeinsam genutzten Knoten bis hin zu bösartigem Verhalten ausgelöst werden können. Die BFT-Knoten einigen sich über den Inhalt des nächsten Blocks, wobei sie alle Eingaben vom unzuverlässigen Bus aufnehmen und eine konsistente Aufzeichnung trotz potenziell willkürlichem Verhalten erhalten. Die geordneten Nachrichten werden in einer Blockchain gespeichert, sodass Löschen von Daten oder Manipulation der Blockchain-Kopie auf allen Knoten ohne Entdeckung unmöglich macht. Sobald ein bestimmter Schwellenwert an geordneten Anfragen erreicht ist, bündeln die Replikate diese deterministisch, versehen sie mit einem Hashwert und speichern

den so erzeugten Block auf der Festplatte. Dies erfüllt die Anforderungen von R3. Dies ist besonders nach Unfällen relevant: Nach der Bergung der ZugChain-Knoten sollte es zu keinem Zeitpunkt möglich sein, selektiv Ereignisse unentdeckt zu löschen, umzuordnen oder anderweitig zu verändern, was durch die Blockchain erreicht wird. Außerdem wird für jeden Block ein *Checkpoint* des BFT-Protokolls erstellt, der somit durch $2f + 1$ asymmetrische Replikat-Signaturen gesichert ist. Auch ein verbleibender ZugChain-Knoten verhindert unentdeckte Veränderung von aufgezeichneten Anfragen. Wenn alle Kopien verloren gehen oder sich in der Kontrolle eines Angreifers befinden, so kann dieser die aktuellsten Daten oder sogar die gesamte Aufzeichnung löschen; dies ist jedoch identisch für die JRU und die Wahrscheinlichkeit, dass mehrere ZugChain-Knoten zerstört werden, ist ausreichend gering. Während die Knoten die relevanten Signale vom Bus ablesen, werden die Konsensnachrichten über eine sekundäre Kommunikationsverbindung, z. B. Ethernet, ausgetauscht, die in Zügen zunehmend verbreitet ist. Dies beugt Änderungen am Kommunikationsbus vor, greift nicht in die bestehende, sicherheitskritische Kommunikation ein und ältere Züge können leichter nachgerüstet werden.

Die Blockchain ist eine *permissioned* Blockchain, wobei die Teilnehmer die Zuggeräte sind und kein unautorisiertes Gerät dem Netzwerk beitreten kann. Die Blöcke werden auf den Knotenpunkten nach Einigung durch das BFT-Protokoll erstellt und gespeichert, bis sie sicher in die privaten Rechenzentren der Eisenbahnunternehmen exportiert werden können. Anders als bei traditionellen BFT-Systemen, bei denen die Clients eindeutige Anfragen stellen, lesen die ZugChain-Knoten weitgehend identische Daten vom Bus, wobei Daten aber übersprungen oder umgeordnet werden können. Im Folgenden beschreiben wir eine Kommunikationsschicht für BFT-Protokolle, die solche Duplikate filtert, um die Systemlast zu verringern und gleichzeitig sicherzustellen, dass keine Informationen im Protokoll ausgelassen werden und somit R1 erfüllt wird.

Um die Daten persistent zu speichern, exportieren wir die Blockchain-Daten. Der kontinuierliche Export von aufgezeichneten Daten während des Zugbetriebs vereinfacht den Extraktionsprozess, ermöglicht Mechanismen zur vorausschauenden Wartung und ist aufgrund des begrenzten Speichers der Knoten erwünscht. Der Zug ist über externe Gateways mit einer Datenverbindung zur Landseite verbunden, z. B. über LTE oder WLAN, deren Abdeckung sich stets verbessert. Da sich die konkurrierenden Bahngesellschaften gegenseitig misstrauen, ist es nicht wünschenswert nur eine Kopie der Blockchain zu haben. Stattdessen werden mehrere synchronisierte Aufzeichnungen in unabhängigen, privaten Datenzentren geführt, die jeweils von einem Unternehmen betrieben werden. Sobald der Zug eine Datenverbindung aufbauen kann, stellen die ZugChain-Knoten eine Verbindung zu einem Datenübertragungsendpunkt auf, der die Knoten auffordern kann, alle Daten zu senden die seit dem letzten Export erstellt wurden. Unser Protokoll stellt sicher, dass nur korrekte Blöcke von den Datenzentren akzeptiert werden und dass die letzten Blöcke tatsächlich exportiert und anschließend gelöscht werden. Dazu fragen Datenzentren die Knoten unabhängig voneinander ab, synchronisieren die Daten und bestätigen den Export, um das Löschen von Blöcken auf den Knoten

anzustoßen. Die privaten Datenzentren speichern die Blockchain-Daten dauerhaft, wodurch ein schneller, unkomplizierter Zugriff auf die gesammelten Daten für die vorausschauende Wartung möglich ist. Das Protokoll für Export und Blockchain-Bereinigung erfüllt R4.

4.3.2 ZugChain für byzantinische Fehlertoleranz

In unserer Implementierung *ZugChain* ist eine Erweiterungsschicht für *Primary*-basierte BFT-Protokolle, die die traditionellen Client-Interaktionen durch eine optimale Verarbeitung von Eingaben über die Buskommunikation ersetzt. Sie stellt sicher, dass alle Bussignale, die an einem korrekten Knoten empfangen werden, erfasst werden, während der Mehraufwand der mehrfachen Erfassung identischer Eingaben durch Filterung vermieden wird. Bei divergierenden Eingaben stellt ZugChain sicher, dass Anfragen, die auf einzelnen Knoten empfangen werden, ebenfalls aufgezeichnet werden und dass jede Anfrage in Verbindung mit der ID eines Knotens aufgezeichnet wird, der sie tatsächlich erhalten hat. ZugChain verwendet eine Schnittstelle, gezeigt in Abbildung 4.4, wie sie typischerweise von einem *Primary*-basierten BFT-Protokoll bereitgestellt wird. Zusätzlich wird die Wahl des *Primarys* und der Fehlerverdacht explizit freigegeben und ermöglicht dadurch die Implementierung der Filterung auf dem *Primary* und das Auslösen von Fehlerverdächtigung des *Primarys*, sollte dieser nicht korrekt filtern. Abbildung 4.4 zeigt auch die Schnittstelle der ZugChain-Knoten, die Daten vom Bus empfangen und die vom BFT-Modul angeordneten Anfragen, die vom BFT-Modul an die Blockchain gesendet werden. Die Interaktionen der ZugChain-Knotenkomponenten sind in Abbildung 4.5 dargestellt.

Module	Call	Explanation
(1) down	PROPOSE(r)	proposes request to consensus group
(1) down	SUSPECT(id)	suspect node to be faulty, init. view change
(1) up	DECIDE(r, sn)	totally ordered request and seq.no.
(1) up	NEWPRIMARY	returns new primary after view change
(2) down	RECEIVE(req)	read parsed request from bus
(2) up	LOG(req, id, sn)	append request to totally ordered log

Abbildung 4.4: Schnittstelle des ZugChain-Protokolls

Der ZugChain-Algorithmus ist dargestellt in Abbildung 4.6. Der Algorithmus kombiniert inhalts- und *Primary*-bewusste Filterung von Anfragen, wobei doppelte Anfragen anhand ihrer Nutzdaten gefiltert werden und Backup-Knoten vermeiden, Duplikate zu übermitteln. Diese Filterung basiert auf dem Protokoll der zuvor entschiedenen Anfragen sowie auf der Warteschlange R, die alle offenen und in Bearbeitung befindlichen Anfragen enthält. Wenn ein fehlerhafter *Primary* Duplikate vorschlägt, wird dies bei der Protokollprüfung erkannt, was zu einem Verdacht und

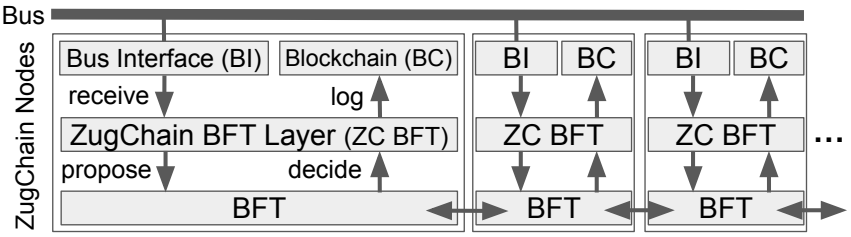

Abbildung 4.5: Interaktionen der ZugChain-Knotenkomponenten

einem Wechsel des *Primarys* führt. Diese Filterung senkt den Einigungsaufwand und gewährleistet dennoch eine vollständige Aufzeichnung. Die Filterung ist eine Optimierung und beeinträchtigt nicht die Korrektheit des Protokolls. Außerdem überwacht ZugChain Zeitbeschränkungen, um eine zügige Protokollierung zu gewährleisten und verlorene oder ausgelassene Anfragen zu verhindern, z. B. stellen Backups nur dann Anfragen, wenn über ihre Eingaben nicht innerhalb einer bestimmten Zeit entschieden wurde. Die Knoten empfangen die Anfragen über den Bus und führen ihre eigenen Warteschlangen (Z. 5, 6). Backup-Knoten starten einen `SOFT_TIMEOUT`-Timer für jede empfangene Anfrage (Z. 11), während der ZugChain-Knoten, der mit dem aktuellen BFT-*Primary* (im Folgenden als (ZugChain-)*Primary* bezeichnet) die erste Anfrage in seiner Warteschlange signiert und `PROPOSE` (Z. 9) aufruft. Sobald das BFT-Modul eine bestellte Anfrage über den DECIDE-Aufruf liefert, prüfen die ZugChain-Knoten, ob sie die entsprechende Anfrage in ihrer Warteschlange haben. In diesem Fall entfernen sie entfernen sie diese und brechen ihren `SOFT_TIMEOUT` ab (Z. 14, 16). Als nächstes prüft das Protokoll auf Duplikate und der primäre Knoten wird verdächtigt, wenn Duplikate entdeckt werden (Z. 17). Wenn der `SOFT_TIMEOUT` eines ZugChain-Knoten abläuft, wurde die entsprechende Anfrage bisher nicht vom *Primary* vorgeschlagen. Dies bedeutet, dass der Knoten in diesem Fall diese Anfrage vorschlägt, indem er die Anforderung an alle Replikate sendet (Z. 21–24). Die ZugChain-Knoten starten dann einen `HARD_TIMEOUT`-Timer und leiten die Anfrage an den *Primary* weiter (Z. 23, 32). Wird die Anfrage innerhalb des `HARD_TIMEOUT`-Timers entschieden, brechen die Backups diesen Timer ab (Z. 16). Andernfalls vermuten die Backups, dass der aktuelle *Primary* fehlerhaft ist, was zu einem Wechsel der aktuellen *View* und schließlich zu einem neuen *Primary* führt (Z. 35). Nachdem ein neuer *Primary* eingerichtet wurde (Z. 36), ruft der kollokierte ZugChain-Knoten `PROPOSE` auf für alle seine offenen Anfragen, d. h. Anfragen ohne eine entsprechende DECIDE oder laufende Konsensinstanz (Z. 41), während die anderen Knoten ihre *SOFT_TIMEOUT* neu starten (Z. 43).

Der ZugChain-Algorithmus stellt sicher, dass alle Anfragen von korrekten Knoten konsistent und mit entsprechenden IDs aufgezeichnet werden. Außerdem werden, falls der *Primary* korrekt ist, alle Duplikate gefiltert, bevor sie an das BFT-Protokoll weitergeleitet werden, was den Einigungsaufwand für die BFT-Anfragen über die Buskommunikation reduziert. Wenn der *Primary* fehlerhaft ist, werden die Duplikate erkannt, der *Primary* wird verdächtigt und schließlich gewechselt (Z. 17). Dies wird durch Überprüfung der Aufzeichnung der zuvor entschiedenen

Algorithm 1 ZugChain Communication Layer

```
1: function Init
2:     R ← ∅                                              ▷ ZugChain node request queue
3:     id ∈ ℕ                                             ▷ ZugChain node id
4:     primary ∈ N                                        ▷ initial primary

5: upon Receive(req_i) do                                 ▷ read parsed data from bus
6:     insert(R, req_i)
7:     if (id == primary) ∧ (!inLog(req_i)) then
8:         r ← sign(req_i, id)                            ▷ authenticate and include (primary) node id
9:         Propose(r)                                     ▷ propose to BFT if co-located with primary
10:    else
11:        t[req_i] ← timer.start(SOFT_TIMEOUT)

12: upon Decide(r, sn) do                                 ▷ receive ordered request returned from BFT layer
13:    if r.req ∈ R then
14:        delete(R, r.req)
15:    if ∃ t[r.req] then
16:        cancel t[r.req]                                ▷ cancel hard or soft timeout
17:    if inLog(r.req) then                               ▷ primary has submitted duplicate request
18:        Suspect(primary)                               ▷ initiate view change
19:    else
20:        Log(r.req, r.id, sn)                           ▷ append to log, include id of origin node

21: upon SOFT_TIMEOUT t[req] expires do
22:    r ← sign(req, id)                                  ▷ authenticate and include node id
23:    t[req] ← timer.start(HARD_TIMEOUT)
24:    Broadcast(r)

25: upon Broadcast(r) do
26:    if inLog(r.req) then
27:        return                                         ▷ ignore duplicates already in the log
28:    if (id == primary) ∧ (r.req ∉ R) then
29:        Propose(r)                                     ▷ propose with id of broadcasting node
30:    else
31:        t[r.req] ← timer.start(HARD_TIMEOUT)
32:        forward r to primary                           ▷ ensure primary receives request

33: upon HARD_TIMEOUT t[r] expires do
34:    if !inLog(r) then
35:        Suspect(primary)                               ▷ initiate view change

36: upon NewPrimary(pid) do                               ▷ after view change
37:    primary ← pid
38:    for ∀ req ∈ R do                                   ▷ for all open requests
39:        if (id == primary) ∧ (!inLog(req)) then
40:            r ← sign(req, id)                          ▷ authenticate and include (primary) node id
41:            Propose(r)                                 ▷ propose on new primary
42:        else
43:            t[req] ← timer.start(SOFT_TIMEOUT)         ▷ reset timers
```

Abbildung 4.6: ZugChain-Algorithmus

Anfragen erreicht. In der Praxis ist eine Überprüfung der gesamten Blockchain für jede Anfrage nicht möglich; stattdessen prüfen wir die jüngste Geschichte. Dies wird effizient mit einer Hashmap über die Anfragen eines gewissen Abschnitts vergangener Prüfpunkte sowie offener Anfragen in R umgesetzt. Der `SOFT_TIMEOUT` ist eine Optimierung, um die zusätzliche Belastung der Einigung von Duplikaten auf dem System zu vermeiden und erlaubt es, Anfragen zu filtern, die auf mehreren Knoten eingegangen sind. Der `HARD_TIMEOUT` wird verwendet, um zu erkennen, ob ein fehlerhafter *Primary* Anfragen unterdrückt und ist mit den Timern, die z. B. von PBFT eingesetzt werden, vergleichbar. Wie in anderen Arbeiten muss der `HARD_TIMEOUT` an die tatsächliche Netzverzögerung angepasst werden, um eine falsche Verdächtigung von korrekten *Primarys* zu vermeiden. Als Optimierung für ein zugrundeliegendes BFT-Protokoll wie PBFT können die ZugChain-Knoten bereits die *Preprepare*-Nachricht als Indikator dafür verwenden, dass die entsprechende Anfrage bearbeitet wird und die entsprechende weiche Zeitbeschränkung abbrechen.

Über die Konsensnachrichten von geordneten Anfragen werden in regelmäßigen Abständen *Checkpoints* (dt. etwa Zwischenspeicherungen) durch das BFT-Protokoll erstellt. Replikate tauschen diese *Checkpoints* der Anwendung in signierten *Checkpoint*-Nachrichten aus und erstellen einen stabilen *Checkpoint*, sobald $2f + 1$ entsprechende Nachrichten empfangen wurden. Da wir jegliche Änderung von Blöcken verhindern wollen, insbesondere nach Abstürzen, wenn nur noch ein Replikat intakt ist, erzeugen wir häufige *Checkpoints*. Ein Block wird erstellt, wenn genügend Anfragen geordnet wurden und für jeden Block wird ein *Checkpoint* erstellt, der diesen Block und alle seine Anfragen enthält. Die signierte *Checkpoint*-Nachricht eines Replikats bestätigt somit, dass es diesen Block erstellt hat. Wir nutzen *Checkpoints* auch für den sicheren Datenexport.

Bisher wurden nur Knoten betrachtet, die an einen einzigen Bus angeschlossen sind. Sie können jedoch auch mit mehreren Eingangsquellen verbunden sein, die möglicherweise (teilweise) synchron sein können. ZugChain kann auch dies effizient handhaben: Knoten haben eine Warteschlange pro angeschlossenen Bus, aus der die Anfragen verarbeitet werden, sodass alle Nachrichten von allen Eingangsquellen aufgezeichnet werden.

4.3.3 Diskussion von möglichem Fehlverhalten

Gemäß unserem Fehlermodell kann sich ein fehlerhafter ZugChain-Knoten, der nicht der *Primary* ist, auf die fünf folgenden Arten falsch verhalten. (i) Anfrageduplizierung: Der fehlerhafte Knoten sendet doppelte Anfragen. Wenn diese bereits im Protokoll enthalten sind oder in Bearbeitung sind, werden sie von allen Replikaten herausgefiltert (über die Funktion inLog, siehe Z. 26). Andere Anfragenduplikate werden von dem *Primary* gefiltert (Z. 28). Wenn diese Duplikate noch nicht noch nicht Teil der Aufzeichnung sind, werden entsprechende harte Zeitbeschränkungen (Z. 31) beim Empfang der DECIDE-Nachricht des Originals abgebrochen (Z. 16). (ii) Auslassen von Anfragen: Anfragen, die nur bei fehlerhaften Knoten empfangen werden, können ausgelassen werden. Dies ist identisch mit einer verlorenen Busverbindung und in Übereinstimmung mit dem JRU-Verhalten. In ZugChain protokollieren wir jedoch weiterhin die Eingaben der korrekten Knoten. Typischerweise werden Anfragen auf allen oder mehreren Knoten empfangen. (iii) *Denial of Service*: Ein fehlerhafter Knoten kann eine große Anzahl von Anfragen senden, um die Leistung des Systems durch Überlastung zu verschlechtern. Um dies zu vermeiden, begrenzt ZugChain die Anzahl der offenen Anfragen, die ein Knoten parallel senden kann, und andere korrekte Knoten verwerfen alle weiteren empfangenen Anfragen. Die Begrenzung wird auf der Grundlage der Busfrequenz berechnet. (iv) Fehlerhafte Übertragungen: Beim Broadcasting (Z. 24) kann ein fehlerhafter ZugChain-Knoten die Anfrage nur an eine Teilmenge von Replikaten senden, sodass im schlimmsten Fall (engl. *Worst Case*) der *Primary* ausgelassen wird. Um einen falschen Verdacht auf einen korrekten *Primary* zu vermeiden, leiten Backups die Anfrage weiter (Z 32). (v) Fehlerhafter Verdacht korrekter *Primarys*: Ein fehlerhafter Knoten kann jeden beliebigen *Primary* verdächtigen. Dies ist nicht problematisch, da die BFT-Protokolle den *Primary* erst dann ändern, wenn er von mindestens $f + 1$ Knoten verdächtigt wurde.

Zusätzlich zu den oben genannten Punkten kann ein fehlerhafter ZugChain-*Primary* möglicherweise Anfragen auslassen oder verzögern oder Duplikate falsch filtern. Das Auslassen von Nachrichten wird erkannt, löst einen Verdacht und eine Änderung des *Views* aus. Die Verzögerung von *Propose*-Nachrichten kann weiche Zeitbeschränkungen verletzen, die zu zusätzlichen Übertragungen führt und somit das System belastet, aber kein Fehlverhalten darstellt. Bei wiederholtem Auftreten könnte dies ebenfalls erkannt werden und zu einem *View Change* führen. Schließlich wird eine fehlerhafte Filterung und das Vorschlagen von Duplikaten durch die Überprüfung der Aufzeichnung bei DECIDE (Z. 17) erkannt, was zu einem *View Change* führt. Wenn in der Praxis die ursprüngliche Anfrage zu einem Duplikat nicht im betrachteten Bereich der *Checkpoints* verfügbar ist, dann wird dieses Duplikat aufgezeichnet. Dies kann zu einer vorübergehenden Leistungsverschlechterung führen; es verletzt jedoch nicht die Korrektheit des Protokolls und diese Doppelung kann bei der nachträglichen Analyse erkannt werden.

4.3.4 Sicherer Rechenzentrumsexport für Blockchain-Daten

Neuere JRU-Daten sind von höherem Interesse und ältere Daten können bei Speicherknappheit verworfen werden, ohne dass die aktuellen Informationen verloren gehen. Da wir jedoch eine Blockchain verwenden, benötigen wir die vollständige Kette, um ihre Integrität zu überprüfen. Daher müssen wir Daten extrahieren, bevor Speicherknappheit oder Datenverluste auftreten. ZugChain kann Blöcke in ein oder mehrere private Datenzentren exportieren, die von den Bahnunternehmen bereitgestellt werden, die bereits JRU-Aufzeichnungen speichern. Jedes Datenzentrum kann neu erstellte Blöcke von den Replikaten anfordern, die dann zwischen allen Rechenzentren synchronisiert und von allen verifiziert werden. Dieser Export ermöglicht es uns, den Zustand der Blockchain zu sammeln und zu verhindern, dass er unbegrenzt wächst. Der Zug ist gelegentlich über eine Datenverbindung mit den Datenzentren verbunden. Da die Bandbreite der drahtlosen Verbindung begrenzt ist, minimieren wir die übertragenen Daten. Anstatt Blöcke von mehreren Knoten anzufordern, nutzen wir das BFT-*Checkpointing*. Jeder Block wird in einen *Checkpoint* aufgenommen und ein stabiler *Checkpoint* mit seinen $2f + 1$ Replikatsignaturen beweist, dass der entsprechende Block in der Blockchain enthalten ist. Da stabile *Checkpoints* nicht mehr Teil des aktiven Anwendungsstatus sind, können wir den Konsens umgehen und die Replikate direkt abfragen. Fehlerhafte Knoten können manipulieren oder über ihre aufgezeichneten Daten lügen, doch die Nutzung stabiler *Checkpoints* stellt die Korrektheit beim Exportieren aus einzelnen Replikaten sicher. Wir fragen zusätzliche Replikate nach ihren *Checkpoints*, um sicherzustellen, dass aktuelle Blöcke tatsächlich exportiert werden und Ressourcen freigegeben werden können. Das Exportprotokoll garantiert, dass (i) nur Blöcke, die von korrekten Knoten aufgezeichnet wurden, exportiert werden; (ii) alle Blöcke bis zum letzten stabilen *Checkpoint* exportiert werden; und (iii) exportierte Blöcke von den Knoten gelöscht werden, um Ressourcen zu sparen. Mit diesem Exportprotokoll erfüllen wir R4.

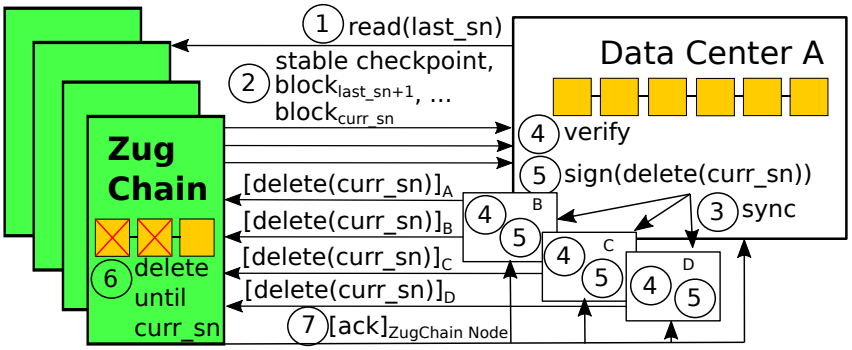

Abbildung 4.7: ZugChain: Ablauf der Kommunikationsschritte

Wir haben zwei Operationen: Lesen, um Blöcke von den Knoten zu extrahieren und Löschen, um einen erfolgreichen Export zu bestätigen, damit die Replikatoren die übertragenen Blöcke sicher löschen können. Alle ZugChain-Knoten sind mit einem asymmetrischen kryptographischen Schlüsselpaar ausgestattet, mit dem sie Einigungs-, *Checkpoint*- und *View-Change*-Nachrichten signieren, was die Überprüfung von *Checkpoints* ermöglicht. Jedes Datenzentrum verfügt ebenfalls über ein Schlüsselpaar, wobei der öffentliche Schlüssel den ZugChain-Knoten bekannt ist und umgekehrt. Der Ablauf der Kommunikationsschritte ist in Abbildung 4.7 dargestellt: (1) Das Datenzentrum fragt die BFT-Replikate nach dem neuesten Block in einem Read Broadcast, der den Index des letzten erfolgreich exportierten Blocks enthält (`last_sn`). (2) Jedes Replikat sendet seinen letzten stabilen *Checkpoint*, während ein zufällig bestimmtes Replikat auch die vollständigen Blöcke (`last_sn` bis `curr_sn`) sendet. Da diese Lesenachrichten den Konsens umgehen und daher ungeordnet sind, d. h. zu verschiedenen Zeiten empfangen werden können, können die Replikate unterschiedliche *Checkpoints* senden. Dies erfordert, dass die Datenzentren den letzten *Checkpoint* mit der höchsten Sequenznummer ermitteln müssen, um die Anzahl der exportierten Blöcke zu maximieren. (3) Sobald die *Checkpoints* von $2f + 1$ Replikaten und die vollständigen Blöcke von einem Replikat empfangen wurden, werden sie mit den Rechenzentren der anderen Unternehmen synchronisiert. Wir warten auf $2f + 1$ Antworten, um den Empfang der letzten *Checkpoints* zu gewährleisten: Während im Prinzip sogar ein einziger gültiger stabiler *Checkpoint* ausreichen würde, könnte dieser *Checkpoint* veraltet sein und daher mehr Daten im Zug hinterlassen als nötig. Mit $2f + 1$ Antworten, selbst wenn f langsame Knoten mit veralteten *Checkpoints* sowie f fehlerhafte Knoten enthalten sind, antwortet mindestens ein Knoten mit einem aktuellen *Checkpoint*, für den wir das Löschen veranlassen können. (4) Unter Verwendung der öffentlichen Schlüssel der Replikate validieren alle Rechenzentren die Signaturen des letzten *Checkpoints* und validieren die empfangenen Blöcke bis zu dem enthaltenen Block. Fehlen Blöcke zwischen `last_sn` und dem im letzten *Checkpoint* enthaltenen Block, können diese direkt von den Replikaten in einer zweiten Kommunikationsrunde abgefragt werden. (5) Nach dem Empfang und der Überprüfung der Blöcke erstellen und signieren die Rechenzentren jeweils eine Löschnachricht, die den Index und den Hash des Blocks im letzten stabilen *Checkpoints* enthält. Die Lösch-Anweisungen werden an die Replikate verteilt. (6) Die Replikate verifizieren nun, dass mindestens eine bestimmte, konfigurierbare Anzahl von Rechenzentren eine signierte Lösch-Anweisung geschickt hat und entfernen die Blöcke bis zu diesem Index, wobei der letzte exportierte Block als erster Block für die gekürzte Blockchain dient. (7) Schließlich senden sie eine signierte Bestätigungsmeldung an die Datenzentren, um die Löschung zu bestätigen.

Im Export können mehrere Fehlerszenarien auftreten: (i) Eine Löschung trifft auf einem Replikat ein, bevor der entsprechende Block erstellt wurde: Das Replikat prüft, ob es den in der Löschung enthaltenen Block erstellt hat. Wenn nicht, verzögert es das Löschen, bis Block und *Checkpoint* beide erstellt wurden, um die Korrektheit zu gewährleisten. Dies könnte vermieden werden, indem die Löschmeldungen über das Konsensprotokoll geordnet werden. Export und Zustimmung

sind jedoch absichtlich entkoppelt, wie es für JRUs erforderlich ist, bei denen der Export die Aufzeichnung nicht verzögern oder beeinflussen sollte. (ii) Übertragen eines *Checkpoints* an ein anderes Replikat: Das Replikat empfängt den *Checkpoint* und blockiert zwischen diesem und dem letzten stabilen *Checkpoint*. Es muss dann die Integrität der Blockchain prüfen und feststellen, ob der letzte Block und die offenen Anfragen mit dem empfangenen *Checkpoint* (bzw. dessen Hashwert) übereinstimmen. Da die Blockchain auf den Replikaten nach einem Export gekürzt wird und die Überprüfung daher nicht beim Entstehungsblock beginnen kann, muss der übertragene Zustand die signierten Löschungen enthalten, die die Basis der Blockchain auf den Replikaten verifizieren. (iii) Nicht genügend Lösch-Anweisungen erhalten: Eine Löschung wird als falsch markiert, wenn das Replikat keine ausreichende Anzahl übereinstimmender signierter Lösch-Anweisungen von den Rechenzentren erhält. Das Replikat führt dann den Vorgang nicht aus. (iv) Ein Datenzentrum ist verspätet und bereits exportierte Blöcke fehlen: In diesem Fall können Blöcke entweder erneut exportiert werden, wenn sie auf den Replikaten noch verfügbar sind, oder von anderen Rechenzentren synchronisiert werden. (v) Ein Replikat verpasst ein oder mehrere Löschvorgänge und gibt den Speicher nicht frei. Bevor irgendwelche Daten aufgrund von Speichererschöpfung überschrieben werden, können die Replikate vereinbaren, die Daten einer bestimmten Anzahl von Blöcken zu löschen und nur ihre Metadaten speichern. Die gemeinsame Vereinbarung wird in der Blockchain gespeichert, um zu signalisieren, dass dies nicht auf ein fehlerhaftes Verhalten zurückzuführen ist. Da die Hashwerte weiterhin zur Überprüfung zur Verfügung stehen, kann das Replikat die Integrität der Blockchain weiterhin überprüfen und gewährleisten. Die signierte Bestätigung der Replikate ermöglicht jedoch, diesen Fall frühzeitig zu erkennen. Dies ermöglicht es dem Wartungspersonal, rechtzeitig einzugreifen. Wir können ferner davon ausgehen, dass die Knoten über ausreichend Speicher verfügen, um aufgezeichnete Daten von mehreren Tagen zu speichern, ähnlich wie bei der JRU.

4.4 Design der Implementierung

Das Einigungsprotokoll ZugChain ist in *Rust* implementiert und kombiniert hohe Leistung mit Speichersicherheit und Effizienz für Geräte mit Ressourcenbeschränkung. *Rust* verhindert undefiniertes Verhalten (z. B. Speicherzugriffe außerhalb des zulässigen Bereichs oder Use-after-free-Fehler), wodurch eine ganze Klasse von möglichen BFT-Fehlern eliminiert wird. ZugChain ist für *Rust* v1.44.0 geschrieben, und die Blockchain-Daten werden via *Protocol Buffers* ausgetauscht. Unser Framework verwendet asymmetrische Kryptographie für die Authentizität von Nachrichten; insbesondere verwenden wir Ed25519-Signaturen für alle Nachrichten. Es umfasst eine vollständige Implementierung von ZugChain und dem klassischen Einigungsprotokoll PBFT, bestehend aus den Teilprotokollen *Ordering*, *Checkpointing*,und *View Change*. Wir verwenden PBFT als das unterliegende Einigungsprotokoll, da es ein etabliertes Protokoll mit gründlich untersuchter Korrektheit ist. ZugChain kann auch andere *Primary*-basierte BFT-Protokolle unterstützen. Unsere PBFT-Implementierung verfügt über die Suspect- und *NewPrimary*-Schnittstellen (vgl. Abschnitt „Spezifikation des Einigungsprotokolls“). Die Exportfunktionalität wird durch die Erweiterung der replizierten Anwendung realisiert. Die Seite der Rechenzentren ist ebenfalls in *Rust* implementiert.

Um auf den Bus zugreifen zu können, enthält unser Framework einen Konnektor zum zugrundeliegenden Bus, in unserem Prototyp der MVB. Der Konnektor hat nur Lesezugriff auf den MVB und verwendet eine spezielle, proprietäre C++-Bibliothek um auf das von Siemens Mobility bereitgestellte Zugkommunikationsnetz zuzugreifen (vgl. Abschnitt „Kommunikationsbus“). Der Datentyp und die Zykluszeit der Signale können dynamisch aus der Buskonfigurationsdatei ermittelt werden, was eine flexible und erweiterbare Handhabung der empfangenen Bussignale ermöglicht. Die Serialisierung der Daten von den MVB-Signalen bis zu den Blöcken und Export-Formaten orientiert sich an den Definitionen der *Protocol Buffers* von *Hyperledger Fabric*. Der Quelltext ist im Internet frei verfügbar [16].

Um möglichst realitätsnah zu sein, haben wir einen Demonstrator mit Hardware aufgebaut, die heutzutage in Zügen eingesetzt wird. Der Aufbau des Testbeds ist in Abbildung 4.8 dargestellt. Es enthält vier Computer vom Typ *M-COM RT V1 QW* (*Modular Communication Router*), die beispielhaft für aktuelle Zug-Hardware sind, die für ZugChain-Funktionen erweitert werden kann, d. h. um den Konsens und die Blockchain auszuführen. Die M-COMs sind mit einer Freescale Quad-Core i.MX 6 Cortex-A9 (ARMv7a) CPU mit 800 MHz, 2 GB RAM, SD-Kartenspeicher sowie GSM-R-Funkfähigkeiten ausgestattet und können zusätzlich zu einer MVB-Verbindung drei 100 Mbit/s und eine 1 Gbit/s Ethernet-Verbindung haben. Auf ihnen läuft ein benutzerdefiniertes Yocto-Linux mit Kernel v3.10.17. Ein DDC, ein Signalgenerator eines JRU-Testsystems, erzeugt Testdaten in der gleichen Form wie der ITC, ein Subsystem der Zugbeeinflussung, und aktualisiert die Signale automatisch. Das verwendete ITC-System ist die PZB. Die DDC entspricht Antennen, die Daten lesen und zur Vorverarbeitung an die ITC übermitteln, bevor die Signale über den Bus gesendet werden und stellt daher einen geeigneten Datensatz für die Auswertung als Teilmenge der relevanten Zugdaten dar. Eine traditionelle JRU, ein

Siemens mRec-s42, ist ebenfalls in den Aufbau einbezogen. Die Komponenten des Testbeds sind an einen MVB angeschlossen, der für ein etabliertes Buskommunikationssystem in heutigen Zügen steht. Dies erfordert einen MVB-Master, der die Buskommunikation regelt, wofür wir einen *SIBAS-KLIP AS318MVB* verwenden. Alle Komponenten sind mit einer Node Supervisor Database (NSDB) Datei ausgestattet, die die Buskommunikation spezifiziert, d. h. welche Signale von der Komponente geschrieben oder gelesen werden. In unserem Aufbau schreibt die DDC Signale, während M-COMs und JRU die übertragenen Signale nur lesen und aufzeichnen. Die M-COMs sind zusätzlich über Ethernet für die Konsenskommunikation angeschlossen. Die DDC und die JRU haben ebenfalls eine Ethernet-Verbindung für Analyse- und Konfigurationszwecke. Ein Netzwerk-Switch verbindet alle Komponenten in einem separaten VLAN und isoliert die Komponenten vom Rest des Netzwerks. Ein LTE-Router stellt eine Verbindung zu einer AWS-VM (t2.xlarge) für den Datenexport her. Ein Raspberry Pi verbindet das Testbed mit der externen Umgebung, um die Experimente zu verwalten und kann auch als Datenzentrum während des Exports dienen.

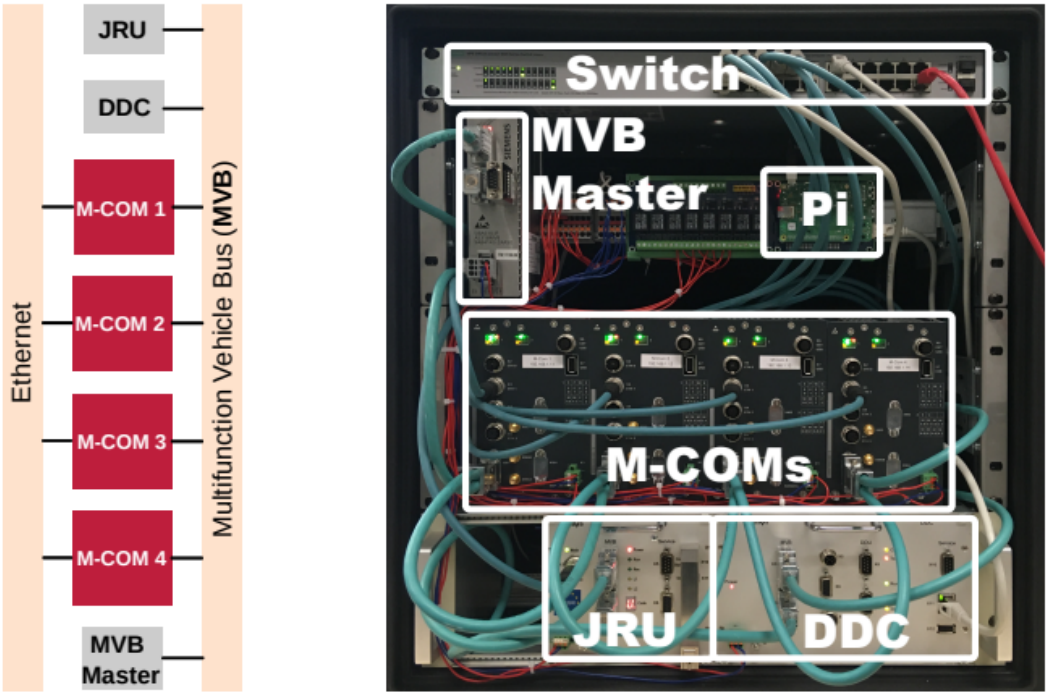

Abbildung 4.8: MVB-Testbed mit M-COM-Rechnern

Die MVB ist ein etabliertes Bussystem für heutige Züge und ZugChain kann leicht an andere Bussysteme für die zukünftige Kommunikation angepasst werden, z. B. auf Basis von *ProfiNet*. Wir können auch zeigen, dass unser Ansatz nicht reaktiv ist, in dem Sinne, dass die übertragenen Signale so aufgezeichnet werden, wie sie sind. Es finden keine weiteren Berechnungen statt, die über die des ITC

hinausgehen, der hier vom DDC simuliert wird. Diese Testumgebung ist daher vergleichbar mit dem Aufbau in einem Zug mit einem MVB und zusätzlichen M-COMs, wodurch wir eine reale Zugumgebung genau simulieren können.

Netzwerkauslastung und Latenzzeit Abbildung 4.9 zeigt die Netzwerkauslastung und Latenz von ZugChain und der Baseline für Buszyklen von 32 ms, dem Minimum der MVB, bis zu 256 ms für eine Nutzdatengröße von 1 kB sowie für Nutzdatengrößen von 32 Bytes bis zu 8 kB bei einem festen Buszyklus von 64 ms. Die Netzauslastung der Baseline für die 100-Mbit/s-Links im Testbed ist vier Mal so hoch wie bei ZugChain, da jede Anfrage vier Mal geordnet und bearbeitet wird. Die Latenzzeit der Baseline ist 1,1–4,9 x so hoch wie die von ZugChain, da mehr Konsensnachrichten übertragen werden müssen. Insbesondere bei einem kurzen Buszyklus von 32 ms sind die Latenzen bis zu 828 x höher. Hier kann die Baseline mit der Anzahl der Nachrichten nicht mithalten und Anfragen bleiben unbeantwortet. Für die Auswertung verschiedener Nutzlastgrößen halten wir den Buszyklus auf den allgemein üblichen Wert von 64 ms fest. Die Latenz von ZugChain steigt um 37 %, während die Latenz der Baseline das 1,6–2,5-fache der Latenz von ZugChain beträgt. Wir benötigen daher weniger Bandbreite und erreichen niedrigere, stabilere Latenzzeiten als die der Basislösung.

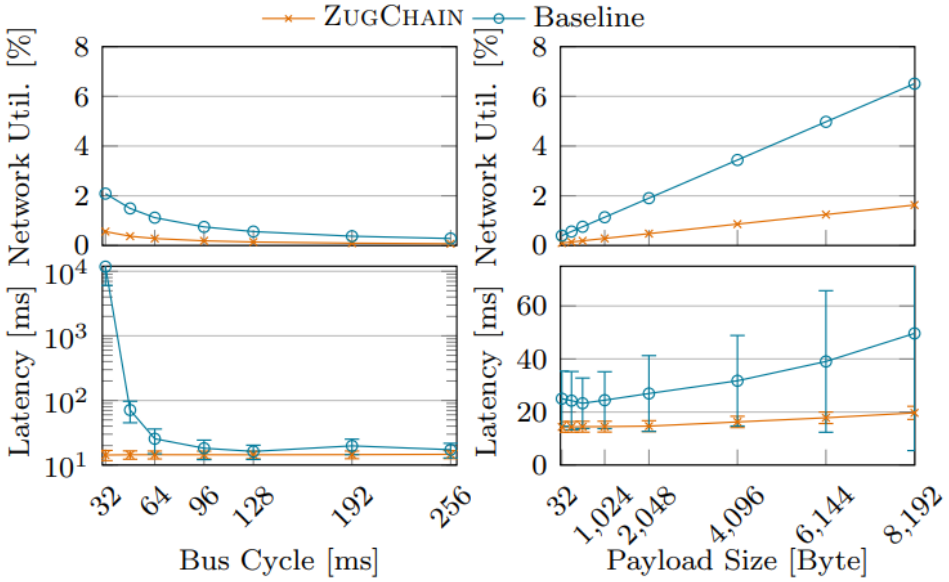

Abbildung 4.9: ZugChain und Baseline im Vergleich: Netzwerkauslastung und Latenz im Verhältnis zu Buszyklen und Nutzdatengrößen

Ressourcenauslastung Abbildung 4.10 zeigt die Speicher- und CPU-Auslastung, wobei 400 % CPU-Auslastung bedeutet, dass alle vier Kerne mit voller Kapazität arbeiten. Die CPU-Auslastung von ZugChain beträgt 25–31 % der der Base-

line für verschiedene Buszyklen und 24–26 % für zunehmende Nutzlastgrößen. Die Speichernutzung der Baseline beträgt das 1,7–1,8-fache der Speichernutzung von ZugChain für verschiedene Buszyklen und das 1,6–1,7-fache für zunehmende Nutzlasten. Bei kurzen Buszyklen benötigt die Baseline bis zu 6,3 x mehr Speicher als ZugChain. Bei einer maximalen Auslastung von 15 % aller verfügbaren CPU-Ressourcen ist ZugChain daher besser für gemeinsam genutzte, ressourcenbeschränkte Standardhardware geeignet.

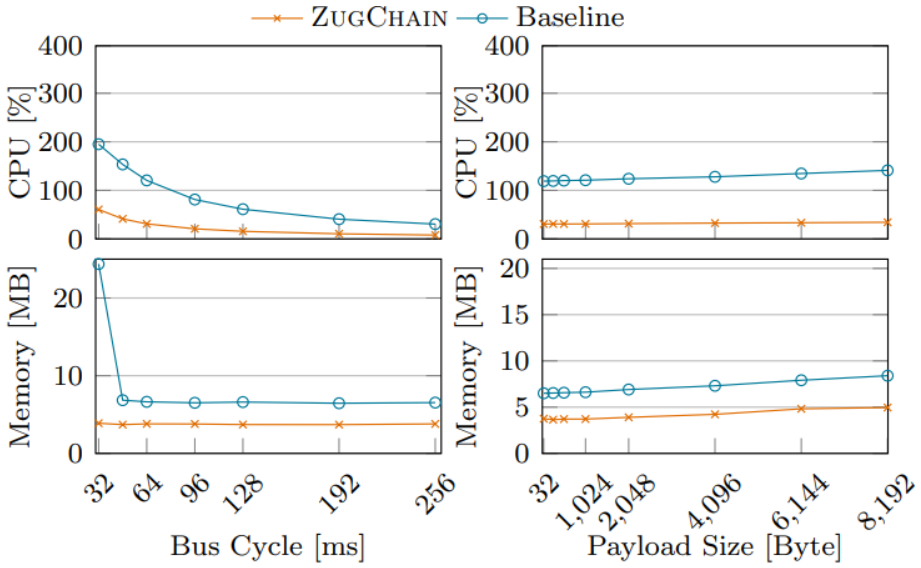

Abbildung 4.10: ZugChain und Baseline im Vergleich: Auslastung von Arbeitsspeicher und Prozessor im Verhältnis zu Buszyklen und Nutzdatengrößen

4.4.1 *View Change*

Abbildung 4.11 zeigt die Anfragelatenz nach einem *View Change* aufgrund eines fehlerhaften *Primarys*, der zum relativen Zeitpunkt Null stattfand. Die Zeitbeschränkung für den *View Change* beträgt bei der Baseline 500 ms, bei ZugChain beträgt die harte und weiche Zeitbeschränkung jeweils 250 ms, insgesamt also 500 ms. Der Buszyklus ist auf 64 ms eingestellt. Das Replikat startet den Timer, sobald er den Fehler entdeckt hat. Nach Ablauf des Timers wird der *View Change* durchgeführt, im Fall von ZugChain 530 ms und für die Baseline 507 ms dauert. Wir haben eine Größe des *Checkpoints* von 10 Anfragen und betrachten den ungünstigsten Fall mit der maximalen Anzahl von Anfragen, die noch nicht in einem Block enthalten sind, also neun Anfragen. Nach dem *View Change* hat die Latenzzeit innerhalb von 210 ms (ZugChain) und 824 ms (Baseline) das vorherige Niveau von 14 ms (ZugChain) bzw. 25 ms (Baseline) erreicht. Wir sehen, dass der *View Change* bei der Baseline etwas schneller abgeschlossen ist, ZugChain sich jedoch schneller stabilisiert, da es weniger Nachrichten zu verarbeiten hat. Die von ZugChain für den *View*

Change benötigte Zeit kann weiter verkürzt werden, je nach den Anforderungen der JRU. Wir optimieren nicht für den Durchsatz, da wir eine feste Anzahl von Nachrichten pro Sekunde haben und ZugChain die nachrichtenintensivste Einstellung, die in unserem Testbed verfügbar ist, ohne Leistungseinbußen bestellen kann (vgl. Abb. Abbildung 4.11). Wir streben stattdessen eine niedrige Latenzzeit an, um eine effiziente Aufzeichnung der Daten zu gewährleisten. Mit unserem sich schnell stabilisierenden *View Change* können wir aggressivere Zeitbeschränkungen verwenden und im Vergleich zur Baseline häufigere *View Changes* akzeptieren, um eine schnelle Wiederherstellung des Normalbetriebs zu gewährleisten.

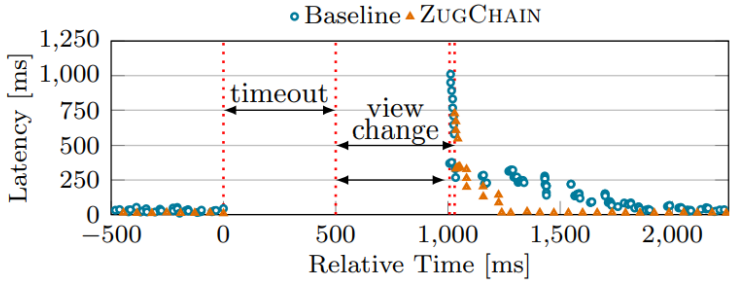

Abbildung 4.11: ZugChain und Baseline im Vergleich: Anfragelatenz vor, während und nach einem *View Change*

4.4.2 Datenexport in Rechenzentren

Der Export besteht aus dem Lesen der *Checkpoints* von $2f + 1$ Replikaten sowie der vollständigen Blöcke von einem Replikat, deren Überprüfung durch das Datenzentrum und deren Löschung. Abbildung 4.12 zeigt die durchschnittlichen Latenzen über fünf Durchläufe dieser Schritte für den Export von 500 bis 16.000 Blöcken an die AWS-VM. Bei einem Buszyklus von 64 ms entspricht dies den gesammelten Daten während eines Betriebs von 5 Minuten bis 3 h. Die Anbindung erfolgt über LTE mit ca. 8,5 Mbit/s. Der größte Teil der Latenz (80–96 %) wird mit dem Warten auf $2f + 1$ Antworten verbracht, insbesondere auf die vollen Blöcke von einem Replikat. Hier ist die Netzwerkkommunikation bis zum Erhalt aller Antworten der Engpass in der Ausführung. Die Verifizierung nimmt 0,2–0,3 % der Gesamtzeit in Anspruch, die Löschung zwischen 3–19 %. Bei einer Dauer von ca. 3 Minuten nach 3 h Betrieb ist der Export von Daten sowohl kontinuierlich oder bei Zugstopps sinnvoll durchführbar.

#blocks	500	1.000	2.000	4.000	8.000	16.000
read	4.9 s	9.8 s	19.7 s	42 s	81 s	162.2 s
delete	0.14 s	0.39 s	4.7 s	9.5 s	12.4 s	15.3 s
verify	0.02 s	0.04 s	0.07 s	0.15 s	0.29 s	0.58 s

Abbildung 4.12: Latenzen des Daten-Exports (Durchschnitt über fünf Durchläufe)

4.4.3 Byzantinisches Verhalten

ZugChain-Knoten können sich fehlerhaft verhalten hinsichtlich (i) des BFT-Protokolls, (ii) des Exports, oder (iii) der Kommunikationsschicht. Fehler gegen das BFT-Protokoll sind ähnlich wie bei anderen BFT-Systemen und führen in der Regel zu einem *View Change*, wenn sie vom *Primary* begangen werden (siehe Abbildung 4.11). Der Export erfolgt nur periodisch und lässt wenig Raum für Fehlverhalten, da mehrere Knoten angefragt werden. Im schlimmsten Fall kann ein fehlerhafter Knoten, der die Antwort verweigert, den Export verzögern, bis ein anderer Knoten angefragt wird. Die kritischsten Angriffe auf die Kommunikationsschicht sind das Hinzufügen verfälschter Nachrichten und die Verzögerung von preprepares durch den *Primary*, um die Leistung des Systems zu verringern. Diese Angriffe werden im Folgenden untersucht, um ihren Einfluss auf das System aufzuzeigen. Andere Angriffe wie das Auslassen oder Verfälschen von Konsensnachrichten führt zu Weiterleitung und potenziellen *View Change*s, doppelte Konsensnachrichten werden auf allen Knoten leicht gefiltert und eine Umordnung von Anfragen ist nicht kritisch. Vom *Primary* ausgelassene Busnachrichten werden weitergeleitet und Duplikate werden wie im normalen Protokollablauf erneut gefiltert. Sowohl die Weiterleitung als auch die Filterung sind Teil der Behandlung von fabrizierten Anfragen und werden im Experiment gezeigt. Abbildung 4.13 zeigt die Ergebnisse dieser byzantinischen Verhaltensweisen: Ein fehlerhafter Backup-Knoten fügt eine gefälschte Anfrage für 25 %, 75 % und 100 % aller Buszyklen. Die Injektion von gefälschten Anfragen führt zu einem Anstieg der CPU-Last um 20 %/68 %/92 %, der Speicherverbrauch um 0,7 %/1,6 %/294 % und die Latenzzeit um 22 %/60 %/277 %, im Vergleich zum Normalbetrieb. Aufgrund der Begrenzung der offenen Anfragen pro Replikat können wir Überlastungen jedoch effizient begrenzen. Die Anfragen werden immer noch innerhalb der Leistungsgrenzen der JRU geordnet, während gutartige Replikate z. B. verzögerte oder nur einmalig empfangene Nachrichten vorschlagen. Abbildung 4.13 zeigt auch einen fehlerhaften *Primary*, der die preprepares um 250 ms verzögert und weiche, aber keine harten Zeitbeschränkungen verletzt, in dem er die Anfragen weiterschickt, bevor ein *View Change* ausgelöst wird. Dies kann den Einigungsvorgang verzögern, bis andere Knoten eine weiche Zeitbeschränkung erhalten und die Anfrage weiterleiten. Dementsprechend steigt die Latenzzeit mit dieser Verzögerung, während die Netzauslastung sinkt. Weiterleitung und weiche Zeitbeschränkungen sind hinsichtlich der Netzwerk- und CPU-Auslastung vernachlässigbar. Dies zeigt, wie wichtig die weiche Zeitbeschrän-

kung ist, die es ermöglicht, die Auswirkungen eines solchen Fehlverhaltens zu begrenzen, um die Systemanforderungen zu erfüllen.

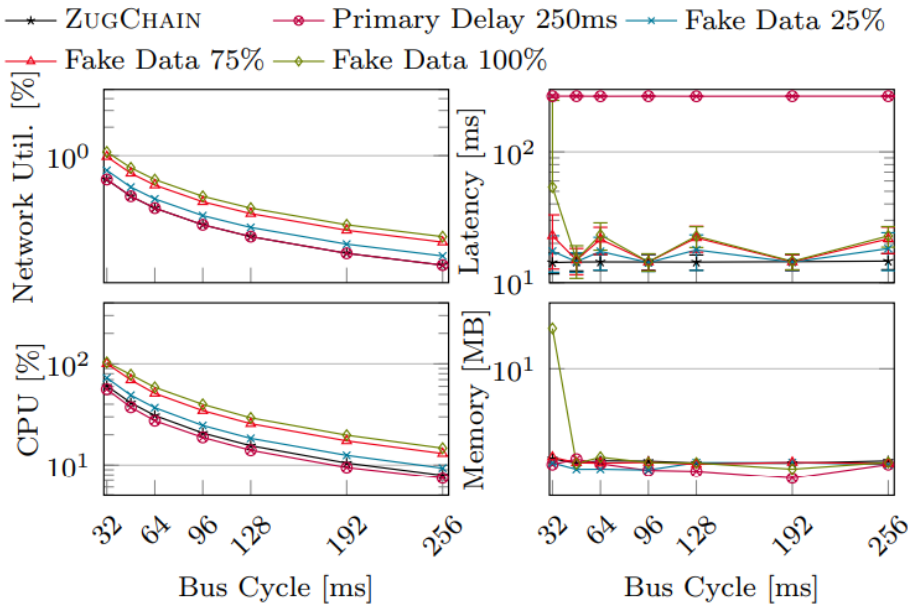

Abbildung 4.13: Auslastung des Netzwerks und des Prozessors im Verhältnis zu Buszyklen und Nutzdatengrößen bei byzantinischem Fehlverhalten

4.4.4 Vergleich mit den JRU-Anforderungen

Ein Datenrekorder muss verhindern, dass Daten gelöscht, verändert oder überschrieben werden und muss die Datenintegrität sicherstellen. Dies erfüllen wir, indem wir Löschungen oder Änderungen erkennen und korrekte Aufzeichnungen auf mehreren Knotenpunkten speichern. Er muss auch die Datenextraktion ermöglichen: Um Datenverlust zu verhindern, exportieren wir häufig Blöcke und um die Datenintegrität nach einem Stromausfall zu gewährleisten, speichern wir die Blockchain persistent auf der Festplatte. Die Daten müssen beim Eintreffen von zehn Anfragen pro Sekunde innerhalb von 500 ms nach ihrem Eintreffen gespeichert werden. Bei einem Buszyklus von 64 ms verarbeiten wir 15,6 Anfragen pro Sekunde, wobei eine Latenzzeit von etwa 14 ms erreicht wird. Das Schreiben von Blöcken auf die Festplatte in unserem Testbed beträgt zusätzliche 5,03 ms bei einer Nutzdatengröße von 8 kB und damit deutlich unter dem Schwellenwert. Wir erfüllen also die JRU-Anforderungen und zeigen, dass ZugChain die Anforderungen von R1 und R2 erfüllt.

5 Fahrzeug-Referenzarchitektur

Nachdem sich die Vorarbeiten mit der Implementation eines Blockchainrekorders im Zug beschäftigten, wurden in diesem Projektteil neue Architekturen analysiert, die bereits grundsätzlich für eine Blockchainintegration geeignet erscheinen. Hier wurden zunächst Architekturen betrachtet, die auch weitere zukunftsgerichtete Innovationen zulassen.

Zugseitig wurde daher zunächst OCORA (*Open CCS On-board Reference Architecture*) [9] analysiert. Diese wird zurzeit von den Betreibergesellschaften DB (Deutsche Bahn), SBB (Schweizer Bundesbahnen), ÖBB (Österreichische Bundesbahnen), SNCF (Société nationale des chemins de fer français) und NS (Nederlandse Spoorwegen) entwickelt. Ziel ist die Bereitstellung einer modularisierten, standardisierten und offenen Zugarchitektur, welche eine Kosteneinsparung für den Betreiber ermöglichen soll. Im Mittelpunkt steht dabei der Einsatz von Hardware die *Commercial Off-The-Shelf* (COTS) verfügbar ist, sodass der Zugausrüster auf weniger spezialisierte Computerhardware zurückgreifen kann. Darauf aufbauend sollen sog. Funktionsblöcke zur Verfügung stehen, die spezielle Teilfunktionen der Funktionen *Control Command and Signalling* (CCS) eines Schienenfahrzeugs bereitstellen. Eine weitere Innovation stellt die Netzwerkkonfiguration dar, welche eine standardisierte Kommunikation der einzelnen Komponenten zulässt. Da hier auf eine Ethernet-Verbindung zurückgegriffen werden soll, stehen somit auch Netzwerkkapazitäten für zukünftige Projekte zur Verfügung. Durch eine redundante Netzwerktopologie mit der Möglichkeit einer Integration von Unternetzwerken, werden auch nichtfunktionale Anforderungen für sicherheitskritische Anwendungen sichergestellt, welche auch zusätzlich von speziellen, sicheren Kommunikationsprotokollen profitieren.

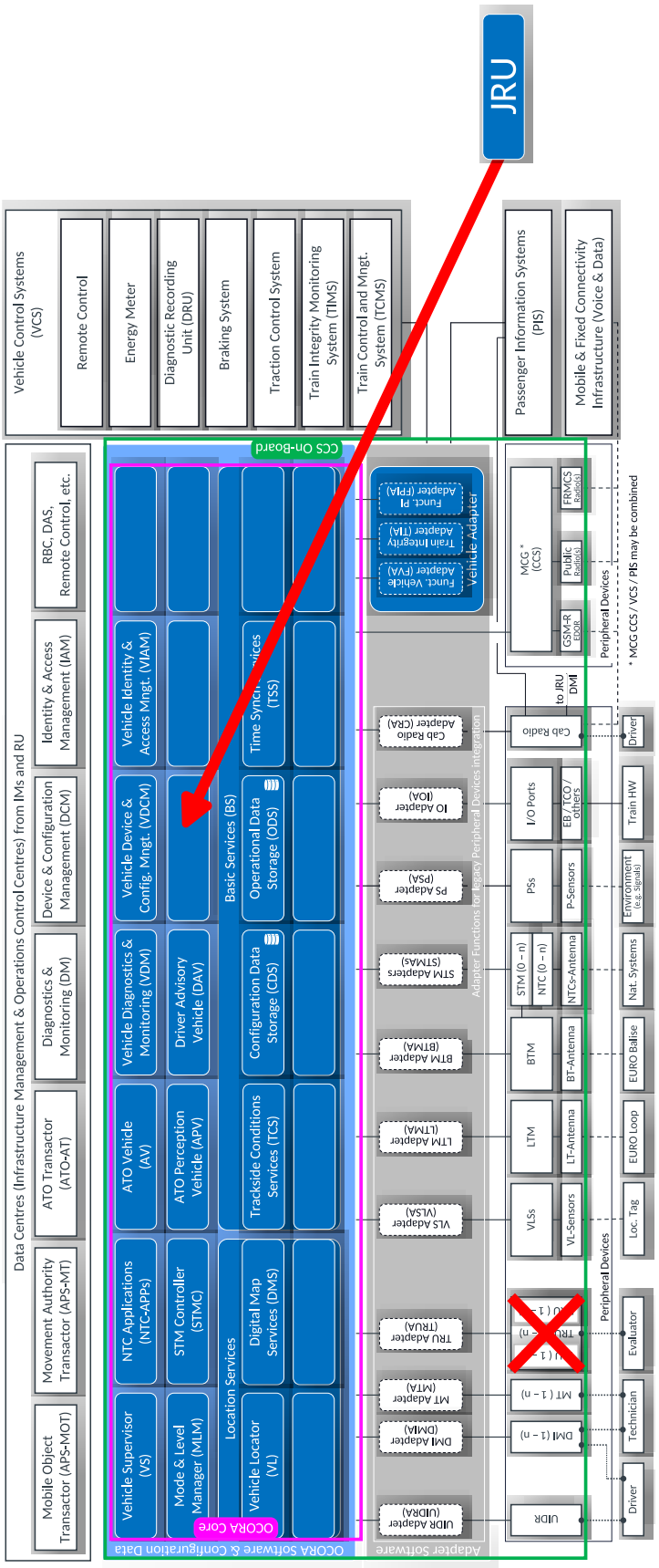

Abbildung 5.1: OCORA-Systemarchitektur aus funktionaler Sicht mit JBR-Modul und gestrichenen Aufzeichnungsperipheriegeräten. [9, bearbeitet]

Das Architekturkonzept OCORA befasst sich mit mehreren, grundlegenden Veränderungen in der Computer- und Netzwerkkonfiguration eines Zuges. Zunächst unterteilt OCORA die Funktionen des CCS eines Schienenfahrzeugs in Funktionsblöcke (z. B. Location Services oder Diagnose), die zunächst unabhängig von der verwendeten Hardware betrachtet werden, siehe Abbildung 5.1. Diese Funktionsblöcke können nun auf vielfältige Weise, einzeln oder gebündelt auf einzelnen oder auf mehreren sicheren Rechenplattformen ausgeführt werden. Die Rechenplattformen werden dabei frei im Netzwerk platziert und über den sog. *Universal Vital Control and Command Bus* (UVCCB) zusammen mit den einzelnen Steuergeräten, wie z. B. dem *Vehicle Control System* angebunden, siehe Abbildung 5.2. Für spezialisierte Peripheriegeräte (z. B. Steuergerät für Balisenantenne, Mobilfunkanbindung) sollen Schnittstellen spezifiziert werden, um eine standardisierte Kommunikation im Netzwerkverbund zu ermöglichen.

Ein weiteres Merkmal der Architektur stellt die Trennung von Hard- und Software dar, die einen einfachen Komponententausch des Betreibers ermöglicht. Diese Modularisierung hat außerdem zur Folge, dass kommerziell verfügbare Hardware eingesetzt werden kann, die nicht durch einen Zulieferer festgelegt wird. Schließlich erlaubt die Modularität einen hohen Flexibilisierungsgrad im Einsatz genutzter Softwareprodukte, da diese durch standardisierte Schnittstellen zur Hardware und zum Bussystem flexibel eingesetzt werden können. Weiteres Ziel von OCORA ist eine leistungsfähige Netzwerkarchitektur, die durch einen hohen Standardisierungsgrad eine leichtere Softwareentwicklung der Zulieferer und eine *Plug-&-Play*-Fähigkeit der Hardware erlaubt. Dieser Architekturentwurf soll somit aus Betreibersicht zu einem erhöhten Wettbewerb unter den Zugausrüstern und neben Kosteneinsparungen durch den Einsatz standardisierter Computerhardware zu neuen Innovationen in der Bahntechnik führen.

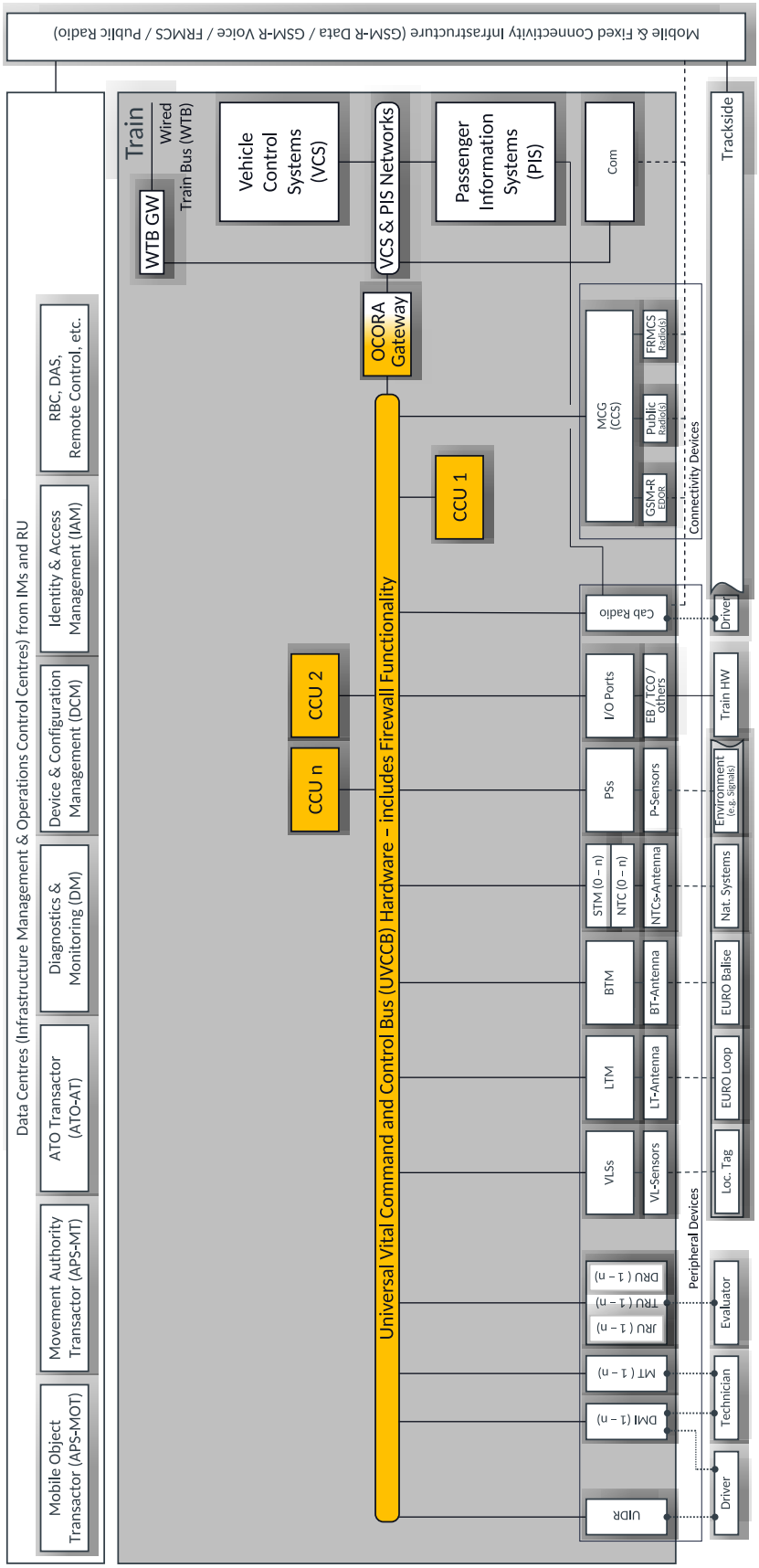

Abbildung 5.2: Hardwarekonfiguration des UVCCBs (gelb) mit angeschlossenen, sicheren Rechenplattformen (CCUs) und Peripheriegeräten (weiß) [9]

Als technischer Mittelpunkt ist das OCORA-Netzwerkkonzept dabei an den *Next Generation Train Control Network* (NG-TCN) Entwurf angelehnt. Hier basiert der physikalische Aufbau zunächst auf dem *Ethernet Train Backbone* (ETB), der durch den gesamten Zug führt und die einzelnen Unternetzwerke der *Consists* über *Ethernet Train Backbone Node*s (ETBNs) verbindet.

Im Konzept für ein Schienenfahrzeug der Zukunft sieht der Entwurf ein einziges physikalisches Netzwerk für einen sog. *Consist* (eine Einheit, z. B. ein Personenzug) vor, bei dem eine Abgrenzung von CCS und *Train Control and Monitoring System* (TCMS) durch unterschiedliche *Virtual Local Area Networks* (VLANs) gewährleistet wird. Hierbei ist das TCMS für die zugindividuellen Steuergeräte, wie Türsteuerung oder Traktion zuständig. Das CCS gewährleistet währenddessen Funktionen für die Übermittlung von Signalinformationen über das Zugbeeinflussungssystem.

Für den *Consist* basiert die Netzwerktopologie bereits auf der NG-TCN empfohlenen Leiter Topologie mit *Dual Homing* (siehe Abbildung 5.3), welches eine erhöhte Ausfallsicherheit der Kommunikation zwischen sicherheitskritischen Geräten bietet, auch wenn für OCORA noch keine detaillierte Analyse der Verlässlichkeit (Zuverlässigkeit, Verfügbarkeit, Wartbarkeit, *Safety* und *Security*) vorliegen.

Für die Bereitstellung der Netzwerkfunktionalität basiert der OCORA Netzwerkentwurf auf einem Ethernet für *Time-Sensitive Networking* (TSN) (mind. 100 Mbit/s, d. h. 100BASE-TX) [38]. Dieses erlaubt eine Zeitsynchronisierung der angeschlossenen End- und Netzwerkgeräte. Zudem können den Verkehrsklassen unterschiedliche Prioritäten bei der Versendung zugewiesen werden. Mittels des *Time-Aware Schedulers* [37] wird die Kommunikation im Netzwerk in feste, zyklische Fenster geteilt, denen ebenfalls eine oder mehrere Prioritätsklassen zugeordnet werden. Dies ermöglicht für Daten hoher Priorität eine garantierte maximale Ende-zu-Ende-Verzögerung, da Kapzitäten im Puffer der Switches reserviert werden. Alle Teilnehmer am Netzwerk wissen durch die Synchronisation, wann Daten welcher Priorität gesendet und verarbeitet werden dürfen.

Somit können zeitkritische Anfragen mit einer Latenz von unter 10 ms (Prozessdaten) bzw. unter 100 ms (Messagedaten) bereitgestellt werden, wobei eine Zeitsynchronisierung der Knoten von +/- 1 ms sichergestellt wird [9]. Als Transport- und Netzwerkschicht wird UDP/TCP sowie IPv4 genutzt. Darauf aufbauend wird das *Train Real-Time Data Protocol* (TRDP) in Fassung 2.0 als Session Layer verwendet. Dieses Protokoll steht (noch nicht finalisiert) quelloffen (TCNOpen) zur Verfügung und erlaubt somit einen standardisierten Datenaustausch von Komponenten unterschiedlicher Hersteller. Für die Sicherungsschicht kommt *Safe Data Transmission* v2 (SDTv2) [33] oder SDTv4 (noch nicht final spezifiziert) zum Einsatz. Dieses Protokoll für sicherheitskritische Aufgaben erlaubt eine Erkennung von Übertragungsfehlern, wie Paketverlust oder Fehlern in der Paketreihenfolge, zur Wahrung von *Safety Integrity Level* (SIL) 2 (SDTv2), sowie SIL 4 (SDTv4).

Die Rechnerplattform von OCORA bietet Mechanismen zur Interaktion mit externen Systemen, bspw. mit Hilfe eines Gateways. Dabei müssen die ausgetauschten Nachrichten durch die funktionale Anwendung per Informationsredundanz (bspw. Paritätsbits, Prüfsummen, *Message Authentication Code*) gesichert werden. Die Nachrichten werden an die Vergleichslogik gesendet und von dort aus an das Gateway

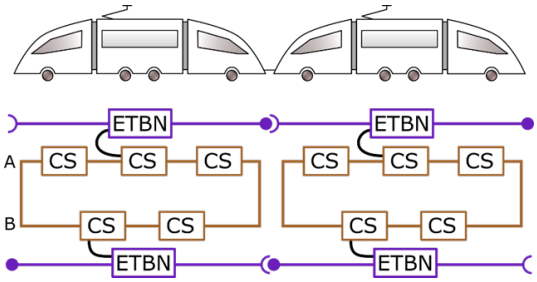

Abbildung 5.3: Netzwerktopologie der *Consists* und des ETB: Endgeräte sind an den *Consist Switches* (CS) angeschlossen, Geräte mit Unterstützung für TSN nutzen zwei CS der Ebenen A und B [8]

weitergegeben. Soll es möglich sein, verschiedene externe Systeme zu adressieren, muss der Nachricht zwischen Anwendung und Gateway eine Identifikation des adressierten, externen Systems zugefügt werden. Die CCUs bzw. die darauf ausgeführten Anwendungen haben somit die Möglichkeit, Daten, die von Peripheriegeräten auf dem UVCCB publiziert werden, zu empfangen. Ebenfalls können sie Daten über den UVCCB senden.

Die Rechnerplattformen sollen harte Echtzeitanforderungen erfüllen. Eine Zeitsynchronisierung der Rechnerplattformen erfolgt dabei durch einen externen Zeitserver. Die aktuelle Zeit wird den funktionalen Anwendungen durch die Rechnerplattformen bereitgestellt. Allerdings wird eine identische Zeit für die zur Fehlertoleranz genutzten Kopien einer Anwendung nur mit einer geringeren Zeitauflösung bereitgestellt.

Der JBR könnte folglich als weiterer Funktionsblock in der OCORA-Architektur berücksichtigt werden (siehe Abbildung 5.1), da hier zum einen auf die bereits vorhandene, moderne Netzwerkarchitektur (Topologie, Bandbreite, Protokolle) zurückgegriffen werden könnte. Zum anderen profitiert das Blockchainkonzept von den sicheren Rechenplattformen, auf denen die Aufzeichnungssoftware ausgeführt werden könnte. Nichtsdestotrotz stellt der Einsatz eines JBR auch Anforderungen an die OCORA-Architektur. So ist für den Blockchainbetrieb der Einsatz von mindestens vier Rechensystemen notwendig. Zudem müssten diese auch über das Schienenfahrzeug hinweg verteilt sein, um Verfügbarkeitsanforderungen ausreichend zu erfüllen.

Es wurde weiterhin in Betracht gezogen, eine Gruppe von Peripheriegeräten zu identifizieren, denen die Aufgabe zugeordnet wird, parallel Daten aufzuzeichnen. Eine Umsetzung des JBR auf Peripheriegeräten wäre jedoch wenig praktikabel. Zum einen soll spezialisierte Hardware, deren Aufgabe allein in der Aufzeichnung von Daten liegt, eingespart werden. Die bisher vorgesehene Hardware zur Datenaufzeichnung durch mehrere, dedizierte Peripheriegeräte (siehe Abbildung 5.1) zu ersetzen, ist demnach nicht sinnvoll. Die Aufzeichnungsfunktion auf Peripheriegeräten parallel ausführen zu lassen, die eine andere Funktion haben, ist ebenfalls wenig vorteilhaft. Zum einen widerspricht dies dem Gedanken der Modularität. Zum anderen müsste die Software des JBR Teil der Spezifikation bestimmter, in

jedem Zug verbauter Peripheriegeräte sein, damit die Blockchain stabil geführt werden kann. Je nach Fehlermodell und gewähltem Konsensalgorithmus ist eine Mindestzahl von Knoten zum Führen der Blockchain notwendig. Wird diese Anzahl unterschritten, kann die Blockchain im Fehlerfall nicht weitergeführt werden. Es müsste demnach eine feste Gruppe von Peripheriegeräten identifiziert werden, der die Aufgabe zugeordnet wird, parallel Daten aufzuzeichnen, was ggf. unnötig die Anforderungen an die Hardware steigen lässt.

Wesentlich praktikabler ist die Umsetzung der JBR auf den sicheren Rechnerplattformen (CCUs) des OCORA-Kerns (*Core*, siehe Abbildung 5.3). Wie beschrieben sind die Rechner bereits dafür ausgelegt, funktionale Anwendungen parallel ausführen zu können. Dabei sollen auch sicherheitskritische und nicht sicherheitskritische Anwendungen auf einer Rechnerplattformen ausführbar sein. Die Rechnerplattformen müssten entsprechend ausreichend Rechenleistung und Speicherplatz zum Führen der Blockchain bereitstellen. Auch wenn die Hardware-Anforderungen der Rechnerplattformen noch nicht spezifiziert sind, ist mit Blick auf die bisher im Testbed des JBR verwendeten *Modular Communication Computer* (MCOMs) davon auszugehen, dass die Anforderungen an die Latenz zum Aufzeichnen der Daten (500 ms nach Erhalt der Daten bei 10 Nachrichten pro Sekunde EN 62625 [35]) erfüllt werden können [50]. Eine Implementierung der JBR auf den sicheren Rechnerplattformen würde jedoch durch rechtliche Vorgaben bedingen, dass das Speichermedium der Rechnerplattform herausnehmbar ist [35].

Abschließend kann die Verwendung der OCORA-Architektur für die Blockchainintegration als sinnvoll betrachtet werden, da hier bereits eine Vielzahl an spezifischen Vorgaben hinsichtlich Rückwirkungsfreiheit, Netzwerk und Rechenplattformen erfüllt werden können.

Perspektivisch soll in der Referenzarchitektur das *Future Railway Mobile Communication System* (FRMCS) eingebunden werden, da spätestens bei der Umsetzung personallosen Fahrens sehr zeitkritische und bandbreitenintensive Daten zwischen Zug und Infrastruktur übertragen werden müssen. Das FRMCS wird dies durch die Nutzung des Mobilfunkstandards der fünften Generation ermöglichen. Dabei soll das FRMCS entweder durch ein *Mobile Communication Gateway* (MCG) mit dem CCS-System verbunden oder direkt im MCG integriert werden, um eine Verbindung des CCS-Systems bspw. mit streckenseitigen Datenzentren herzustellen. Da in der Referenzarchitektur der Speicherplatz zum Führen der Blockchain weiterhin begrenzt sein wird, kann das MCG bzw. FRMCS zur schnellen Übertragung der Blockchain an einen Cloudservice genutzt werden. Analog zur Implementierung im Testbed reicht es dann aus, aktuelle Blöcke der Blockchain vorzuhalten [50].

5.1 Streckenseitige Referenzarchitektur

Streckenseitig wurde eine ähnliche Architekturanalyse durchgeführt. Hier war das Ziel, wie der Blockchainrekorder in eine Stellwerksarchitektur integriert werden könnte, da auch hier der Bedarf besteht, juristisch relevante Daten für den Fall eines Zwischenfalls oder einer Störung aufzuzeichnen. Dies wird wie bei OCORA auch deswegen relevant, weil in beiden Architekturen Komponenten verschiedener Hersteller gemischt werden sollten. Dies birgt die Herausforderung, dass man angefangen von den Integrationstests bis zum Betrieb eine eindeutige, nicht abstreitbare „Datenwirklichkeit“ benötigt, um unnötige Mehrarbeit wie die Synchronisation und Abstimmung der Daten aus verschiedenen Quellen zu vermeiden.

Basis der Analyse war hier die EULYNX-Architektur, welche von 13 europäischen Bahninfrastrukturbetreibern vorangetrieben wird. Ähnlich wie OCORA, bietet auch EULYNX eine modularisierte Architektur mit der Anwendung von Zugsteuerung, Zugsicherung und Signalgebung [2]. Grundlegend bietet dieses Konzept zunächst vier Subsysteme: „Stellwerk“, „Feldelemente“, „*Maintenance and Data Management* (MDM)“ und „Kommunikationssystem“. Im Mittelpunkt der EULYNX-Dokumentation steht dabei die Schnittstellenspezifizierung der einzelnen Systeme und Subsysteme, sodass eine einheitliche Kommunikation über Herstellergrenzen hinweg ermöglicht wird. Für die Kommunikation wird erneut das kapazitätsstarke Ethernet mit speziellen Kommunikationsprotokollen genutzt.

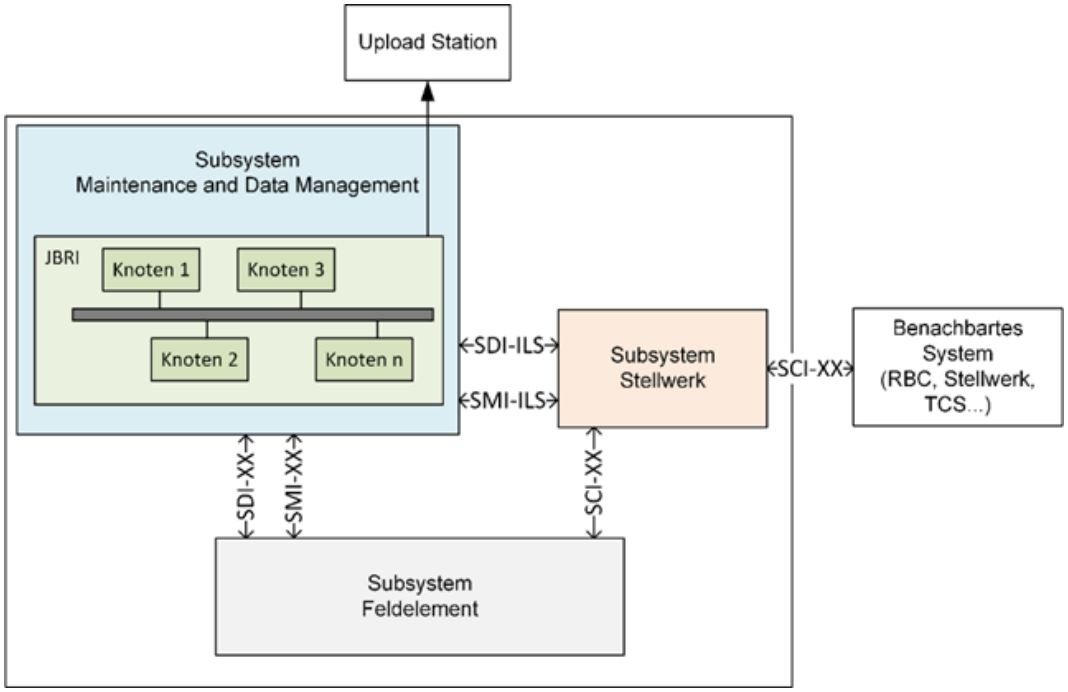

Abbildung 5.4: Adaptierte EULYNX-Architektur, erweitert um Hochladestation und einer JBR-Integration im Subsystem *Maintenance and Data Management*

Zu jedem Subsystem werden nun drei Schnittstellen definiert, die die oberen Schichten des nach dem ISO-OSI-Modell aufgebauten Protokollstapels zur telegrammbasierten Kommunikation darstellen:

1. Prozessdaten-Schnittstelle (*Standard Communication Interface*, SCI-XX) [19, 21]
2. Diagnosedaten-Schnittstelle (*Standard Diagnostic Interface*, SDI-XX) [20, 22]
3. Wartungsdaten-Schnittstelle (*Standard Maintenance Interface*, SMI-XX) [18]

Prozessdaten haben die höchste Priorität, gefolgt von Wartungsdaten. Diagnosedaten haben die geringste Priorität. Für jede Schnittstelle soll die Priorisierung separat nach IEEE 802.1Q [36] definiert werden. Die Dienstgüte (engl. *Quality of Service*) der einzelnen Datenschnittstellen ist in Tabelle 5.1 zusammengefasst.

	SCI-XX	**SMI-XX**	**SDI-XX**
Datendurchsatz	15 kbit/s (CIR) 30 kbit/s (PIR)	45 kbit/s (CIR) 2 Mbit/s (PIR)	30 kbit/s (CIR) 2 Mbit/s (PIR)
Verzögerung	50 ms	250 ms	250 ms
Paketverlustrate (innerhalb von 5 Min.)	< 1%	< 5%	< 5%
Priorität	> SMI-XX/SDI-XX	> SDI-XX	—

Tabelle 5.1: Dienstgüte (engl. *Quality of Service*) der verschiedenen Datenschnittstellen [24] CIR: committed information rate; PIR: peak information rate

Laut EULYNX-Spezifikation soll das Subsystem *Maintenance and Data Management* eine Service-Funktion *Logging* besitzen, mit der die zwischen den Sub- und benachbarten Systemen ausgetauschten Daten aufgezeichnet werden können. Das MDM greift dabei die Prozessdaten, die über die Schnittstelle SCI-XX versendet werden, per repliziertem Port (engl. *mirror port*) oder *Network Terminal Access Point* (TAP) am zentralen Anschluss des Stellwerks am *PoS Signalling* ab [23]. Dadurch stehen im MDM sowohl die Prozess- als auch die Wartungs- und Diagnosedaten zur Verfügung. Diese werden vom *Juridical Blockchain Recorder for Infrastructure* (JBR) als Teil des MDM aufgezeichnet, der somit die geforderte Service-Funktion *Logging* umsetzt. Der JBRI ist ein Netzwerk von Rechnern, die die empfangenen Daten manipulationssicher in einer Blockchain abspeichern. Jeder Knoten des Netzwerks führt eine identische Kopie der aufgezeichneten Daten. Diese können zur langfristigen Speicherung vom JBRI auf eine Hochladestation hochgeladen werden.

Für die Integration einer Blockchain erscheint daher das Subsystem MDM als sinnvoll, da hier bereits grundlegende Aufzeichnungsfunktionalitäten in den EULYNX-Spezifikationen beschrieben werden. Folglich kann ein Teil der Kommunikation bereits im MDM abgegriffen werden. Weiterer Netzwerkverkehr von Prozessdaten kann über replizierte Ports oder TAP mitgeschnitten werden. Es ergibt

sich schließlich die in Abbildung 5.4 dargestellte Architektur, bei der der EULYNX-Entwurf um eine Blockchain inklusive Hochladestation erweitert worden ist.

Weiterhin wurden in diesem Projektteil spezifische funktionale und nicht-funktionale Anforderungen an diese adaptierte Architektur formuliert, sodass eine Realumsetzung auf die vom Projekt formulierten Anforderungen aufbauen kann. Wichtig zu benennen ist, dass viele Prozesse im Softwareengineering auf die Integration einer Blockchain abgestimmt sein müssen, so darf ein einfacher Hardwaretausch die Aufzeichnungsfunktionalitäten nicht beeinflussen. Daher sind erneut eine Vielzahl an Use Cases formuliert worden, um möglichst viele Anwendungsfälle im Produktlebenszyklus abbilden zu können.

5.1.1 Funktionale Anforderungen

Anhand des Use-Case-Diagramms in Abbildung 5.5 können die Beziehungen der mit dem JBRI interagierenden Nutzer bzw. Systeme aufgezeigt werden. Die funktionalen Anforderungen werden vom Verhalten des JBRI gegenüber diesen Akteuren abgeleitet.

Als Beispiel zeigt die folgende Tabelle das grundsätzliche Szenario „Daten aufzeichnen“ für die Infrastrukturseite.

Name	**Daten aufzeichnen**
Kurzbeschreibung	Der JBRI zeichnet die relevanten Daten auf, die über verschiedene Schnittstellen an ihn kommuniziert werden und die er durch Selbsttest oder Datenupload/-Datenentnahme selbst erzeugt.
Akteure	Subsysteme der Infrastruktur (insbesondere Stellwerk, MDM, Feldelemente), benachbarte Systeme (insbesondere ETCS *Radio Block Center*, ETCS-Streckeneinrichtungen, TCS, benachbarte Stellwerke)
Vorbedingungen	Der JBRI besitzt eine Schnittstelle zu den Prozess-, Diagnose- und Wartungsdaten. Die Systeme senden die aufzuzeichnenden Daten an das MDM bzw. den JBRI.
Fachlicher Auslöser	Aufzuzeichnende Daten werden vom JBRI als solche erkannt.
Ergebnis	Aufzuzeichnende Daten sind im Speichermedium des JBRI gespeichert.
Hauptszenario	Der JBRI zeichnet die Wartungs- und Diagnosedaten auf, die das MDM verschickt oder erhält. Ebenfalls werden die vom MDM per repliziertem Port oder TAP abgegriffenen Prozessdaten aufgezeichnet. Der JBRI zeichnet zudem selbst erzeugte Daten zu Selbsttest und Datenupload/Datenentnahme auf.
Alternativszenarien	—

Nachbedingungen	Der JBRI ist rechtzeitig bereit, weitere Daten zu erkennen und abzuspeichern.

Als weitere Use Cases wurden „Datenentnahme“, „Datenupload (Backup)“, „Konsens finden“, „Software aktualisieren“, „Selbstest durchführen“, „Diagnosemeldung“ sowie „Hoch-/Herunterfahren“ beschrieben. Aus den Use Cases wurden dann detaillierte Anforderungen mit eindeutigen Identifikatoren abgeleitet.

Identifikator	`#REQ_JBRI_Daten_0001:02#DEF#`
Titel	**Datenquellen**
Beschreibung	Der JBRI muss die Prozess-, Diagnose- und Wartungsdaten von allen Quellen des Stellwerks, der Feldelemente, des MDM sowie der benachbarten Systeme aufzeichnen.
Ursprung	—
Querbezüge	—
Hinweise	In den benachbarten Systemen eingeschlossen sind: • TCS • Benachbarte Stellwerke • ETCS *Radio Block Center* • ETCS-Streckeneinrichtungen

Nicht-funktionale Anforderungen an Verlässlichkeit (Zuverlässigkeit, Verfügbarkeit, Wartbarkeit, *Safety* und *Security*), die Beständigkeit des Speichermediums, Brand- und Umweltschutz, On-Board-Anwendung und Zulassung wurden ebenfalls detailliert dargelegt.

Identifikator	`#REQ_JBRI_Zuverlaessigkeit_0006:01#DEF#`
Titel	**Klima Betrieb**
Beschreibung	Der JBRI muss die Betriebsfähigkeit unter den klimatischen Umweltbedingungen der Klasse 3K21 nach EN 60721-3-3 [31] gewährleisten.
Ursprung	—
Querbezüge	—
Hinweise	3K22: gilt für temperierte, geschlossene Einsatzorte. Die Luftfeuchte wird dabei nicht geregelt.

Einige spezifische Werte sind bisher nur mit Platzhaltern versehen und müssen noch im Rahmen des Abstimmungs- bzw. Standardisierungsprozesses abgestimmt werden.

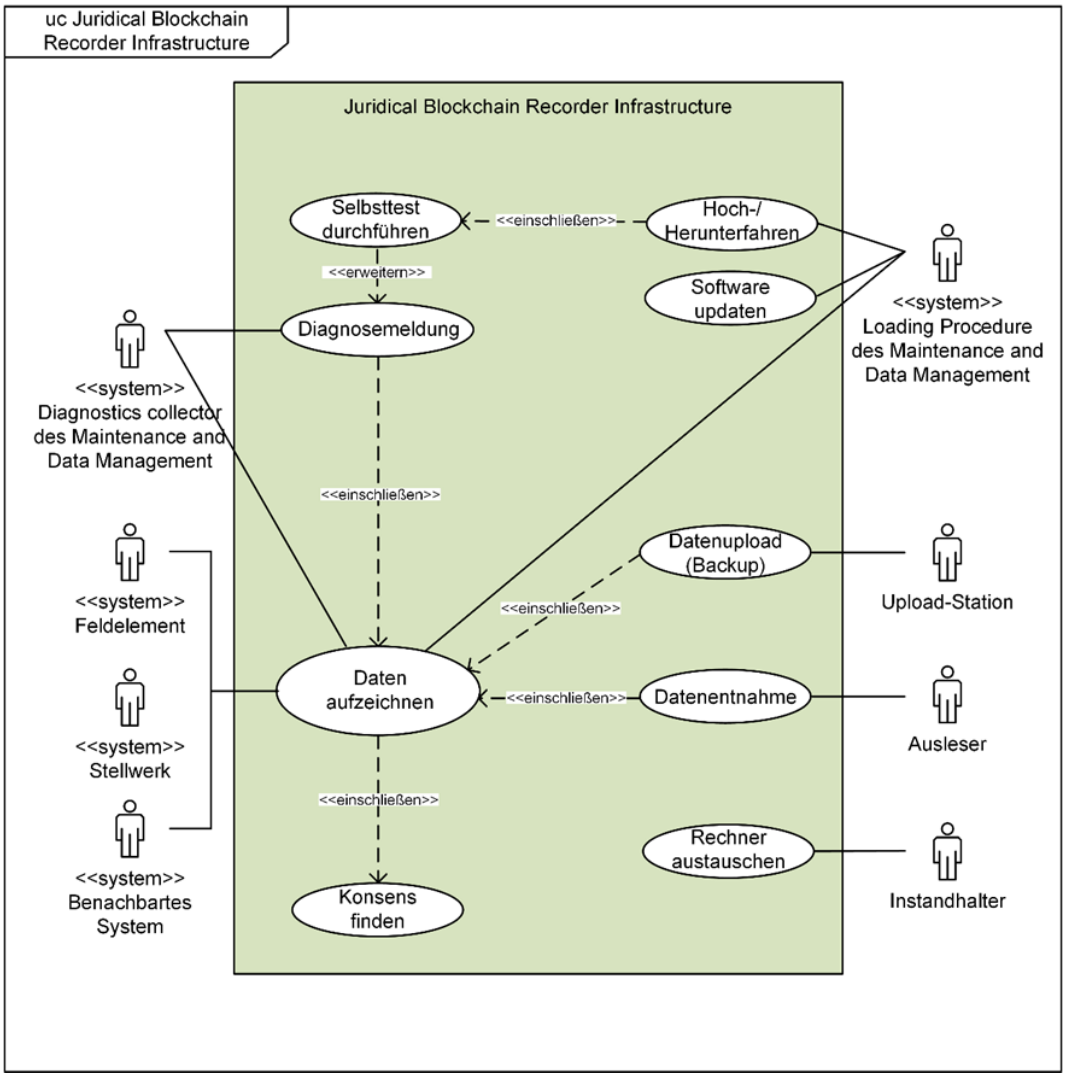

Abbildung 5.5: Use-Case-Diagramm JBR Infrastructure

6 Vermessung in Labor, Simulation und Feld

Ziel der Erstellung des Demonstrators war die Herstellung eines lauffähigen Prototyps zur Evaluation im Labor und auf dem Fahrzeug. Im Folgenden geben wir einen Überblick über die Beschaffenheit, Entwicklung, sowie qualitative und quantitative Evaluation der Prototypen. Dabei untersuchte das Konsortium die Effizienz und die Umsetzbarkeit des vorgeschlagenen Konzepts für die fälschungssichere Datenaufzeichnung im Zug. Zusätzlich fanden die Themen um *RailChain* Einzug in die akademische Lehre. Die Forschungsinhalte wurden Studierenden somit nahegebracht und andersherum konnten Impulse für das Projekt durch die Studierenden aufgenommen werden.

6.1 Beschreibung des Demonstrators

Aus den Untersuchungen zu technischen Grundlagen ist an der TU Braunschweig die Implementierung *ZugChain* hervorgegangen, die im Kern aus einem effizienten Konsensalgorithmus besteht und für die Demonstration und Vermessung mit Schnittstellen zum Einspeisen von Sensordaten sowie einem Cloud-Export erweitert wurde.

In Ergänzung hierzu fokussierte sich das Hasso-Plattner-Institut (HPI) mit dem Projekt *Prellblock* auf den Entwurf einer komponentenorientierten Architektur, die den Austausch von Systemteilen je nach Anforderungen in verschiedenen Anwendungsfällen erlaubt. Beispielhaft wurde hierfür die Konsensfindung auf das vom OCORA-Konsortium favorisierte *Train Real-Time Data Protocol* (TRDP) über *Time-Sensitive Networking* (TSN) portiert.

Prellblock wurde im Kontext eines Bachelorprojekts [29] von den Studierenden Malte Andersch, Benedikt Schenkel, Felix Gohla und Martin Michaelis entwickelt. Der im Kern in der Programmiersparche *Rust* implementierte Prototyp hatte neben der komponentenorientierten Architektur auch Effizienz, Portabilität und Sicherheit (i. S. v. Vertraulichkeit) zum Ziel. Es entstand unter anderem ein für *RailChain* modifiziertes Konsensprotokoll, eine Benutzeroberfläche (*Thingsboard*[1]) für Geschäftsanwendungen aus dem Bereich des *Internet of Things* (IoT), sowie eine grafische Oberfläche (*Grafana*[2]) zur Darstellung von Leistungsmetriken.

[1] `https://thingsboard.io/`, besucht am 23.12.2022
[2] `https://grafana.com/`, besucht am 23.12.2022

Schließlich gab es Anpassungsbedarf für den Betrieb der beiden Softwareprojekte im Versuchsträger *advanced TrainLab*.

Anknüpfungspunkte Im Laufe des Projekts fand fortwährend ein Austausch zwischen Fahrzeugbetreibern (EVUs), Herstellern und Behörden statt. Dadurch erhielten die Teilnehmenden im Verlauf des Projekts auch zunehmende Einblicke in die Bewegungen bezüglich zukünftiger Architekturen, Technologien und Standards.

6.1.1 Referenzarchitektur OCORA

An dieser Stelle sei OCORA (*Open CCS On-board Reference Architecture*) hervorgehoben, die Europäische Initiative zur ETCS-Fahrzeugausrüstung der Zukunft. Das Konsortium entwickelt mit offenen Prozessen und Daten eine Referenzarchitektur für die Fahrzeugseite zukünftiger Leit- und Sicherungstechnik (ETCS). Im Sinne von OCORA könnte *RailChain* in der Referenzarchitektur vorgesehene Komponenten wie dem *Data Recording On-Board* (DR-OB) bereitstellen. Zusätzliche Relevanz für *RailChain* ergibt sich daraus, dass OCORA ebenfalls zugrundeliegende Technologien empfiehlt.

Darüber hinaus entstand die Idee, dass das in *RailChain* entwickelte Konsens-Protokoll auch als eine Middleware-Komponente in OCORA integriert werden könnte. Da auch andere verteilte Systeme auf den Fahrzeugen Daten austauschen, könnte für diese der Implementierungsaufwand für ähnliche Funktionalitäten eingespart werden. Das verteilte, replizierte und vor Änderungen geschützte Austauschen von Daten könnte dabei nicht nur auf der Fahrzeugseite stattfinden, sondern bei Bedarf auch bis auf die Landseite ausgedehnt werden (Cloud-Export, s. o.). Wenngleich diese Idee deutlich über den Rahmen von *RailChain* hinausgeht, ergeben sich daraus interessante Perspektiven. Ein fachlicher Austausch mit an OCORA Beteiligten hat stattgefunden.

6.1.2 Anpassung des Demonstrators für den Versuchsträger

Für Entwicklungs-, Test- und Demonstrationszwecke sollten vier bahntaugliche Industrie-Computer vom Typ *smartRAIL* des Herstellers *optiMEAS* auf dem Versuchsträger installiert werden (siehe Unterabschnitt 7.2.2). Dadurch, dass die Geräte herstellerseitig mit einem Linux-Betriebssystem betrieben werden, sind sie grundsätzlich geeignet um sowohl die ZugChain-, als auch die *Prellblock*-Implementierung auszuführen. Um jedoch den weiteren Anforderungen für den Test- und Demonstrationsbetrieb gerecht zu werden, mussten die Systeme im Kontext des Projekts noch grundlegend angepasst werden.

Die *smartRAIL*-Geräte werden mit einem angepassten Linux ausgeliefert, auf dem beispielsweise nicht ohne Weiteres Software nachträglich installiert werden kann. Allerdings sollten zum einen gängige Software-Werkzeuge zur Protokollierung, Wartung und Fehlersuche bereitstehen, zum anderen musste für den Fernzugriff über Mobilfunk eine VPN-Lösung entwickelt werden. Es war naheliegend, dass

sich die Anforderungen im Projekt durch den Betrieb gängigen Linux-Distribution, wie Debian, auf den Geräten erfüllen ließen. Für *optiMEAS* war die Installation von Debian auf ihren Geräten eine zwar gewünschte, jedoch noch wenig erprobte Alternative. Die Installation von Debian entwickelte sich jedoch zu einer eigenständigen Herausforderung, die in regelmäßigem Austausch mit der Firma *optiMEAS* bearbeitet worden ist.

Um die im Projekt entstandenen Prototypen unter den o. g. Anforderungen entwickeln und betreiben zu können, wurden zunächst Vorarbeiten außerhalb des Versuchsträgers durchgeführt. Hierfür hat die Fa. *optiMEAS* sowohl ein Exemplar der *smartRAIL*-Geräte bereitgestellt, als auch wöchentliche Video-Konferenzen zum Erfahrungsaustausch mit ihren Mitarbeitenden aus der Entwicklungsabteilung ermöglicht. Mithilfe des Probegeräts konnten die Anpassung des Linux-Betriebssystems für die aus *RailChain* hervorgehenden Anforderungen im IoT-Labor (dem Labor für das *Internet of Things*) realisiert und erprobt werden. Nicht nur durch diese etwas neuartige Verwendung der Geräte, sondern auch durch separate, sporadische Evaluationen konnten auch Erkenntnisse aus dem Projekt zurück zur Fa. *optiMEAS* fließen. Beispielsweise wurde die Fehlertoleranz gegenüber Stromausfällen von verschiedenen Dateisystemen über einen Zeitraum von mehreren Tagen experimentell erprobt. Der regelmäßige fachliche Austausch und die letztendlich erfolgreichen Tests mit dem Probegeräte im IoT-Labor des HPI bewies sich als eine Bereicherung für alle Beteiligten, inklusive der Firma *optiMEAS*. Um den Projektpartnern ebenfalls die Vorbereitung der Tests und Demonstrationen unter Laborbedingungen zu ermöglichen, wurde – wie für die späteren Feldtests – eine Lösung für den Fernzugriff entwickelt und bereitgestellt. Die in Zusammenarbeit mit der *optiMEAS* entwickelten Betriebssystemanpassungen wurden samt Dokumentation auf *GitHub* veröffentlicht [15].

Nachdem Erfahrungen mit dem Gerät im Laborbetrieb gesammelt waren, stand der Einbau der letztendlichen Test-Geräte in das *advanced TrainLab* bevor. Bei einem Vor-Ort-Termin bei der DB Systemtechnik in Cottbus wurden die Geräte softwareseitig vorbereitet. Trotz einiger unvorhersehbarer technischer Herausforderungen konnten alle vier für das Protokoll der *Distributed-Ledger*-Technologie (DLT) benötigten Geräte innerhalb schließlich bei einem Termin im Instandhaltungswerk in Halle in Betrieb genommen werden. Die Installationen für die konkreten Tests und Demonstrationen fanden anschließend, wie im IoT-Labor am HPI erprobt (s. o.), per Fernzugriff statt.

Für den verlässlichen Betrieb des Aufbaus sowie für die publikumswirksame Darstellung der Projektthemen setzten wir uns weiterhin mit dem Umgang mit den anfallenden Diagnosedaten (Ringpuffer auf Ebene des Dateisystems), dem Umgang mit Stromausfällen an den Testsystemen (Tolerierung von Systemabstürzen), der Stromversorgung der Demonstrationssysteme an Board (Batteriebetrieb und Lastbegrenzung), sowie einer Ortung des Fahrzeugs (GNSS) und deren Darstellung auseinander (Echtzeitkarte).

6.2 Vermessung und Evaluation

Die in *RailChain* entworfenen Prototypen der *Juridical Blockchain Recorder* (JBR) benötigen ein Kommunikationsmedium auf dem jeweiligen Fahrzeug. Auf bestehenden Fahrzeugen gibt es einen oder mehrere Busse, die die verschiedenen Subsysteme untereinander vernetzen (z. B. *Multifunction Vehicle Bus* (MVB), s. o.). Die Kapazitäten für die Datenübertragung auf solchen Bussen und die Anzahl der Busteilnehmer ist in der Regel limitiert bzw. nicht ohne großen Aufwand änderbar. Gerade unter Berücksichtigung existierender Zulassungen, kann ein System wie *RailChain* im Nachhinein nicht wirtschaftlich in solche bestehenden Kommunikationsmedien eingegliedert werden.

Dank der flexiblen Möglichkeiten auf dem Versuchsträger war die Installation eines separaten Ethernet-Netzwerks für den Betrieb der Prototypen problemlos möglich. Für den dauerhaften und produktiven Produktivbetrieb jedoch wäre ein solches separate Kommunikationsmedium untypisch und ungewollt. Folglich mussten für den zukünftigen Einbau und Betrieb realitätsnahe alternative Strategien evaluiert werden.

6.2.1 Kommunikation per Time-Sensitive-Networking

In Bezug auf ein fahrzeugseitiges Kommunikationsmedium erwies sich während der Recherchen in *RailChain* das Projekt Shift2Rail als besonders relevant. Shift2Rail ist eine europäische Initiative für Forschung, Innovation und marktorientierte Lösungen im Bahnsektor. Durch beschleunigte Integration von neuen Technologien in innovative Lösungen, fördert Shift2Rail den Wettbewerb der europäischen Bahnindustrie und unterstützt die Verkehrswende auf europäischer Ebene. So sind beispielsweise eine Kapazität im europäischen Schienennetz von 200 %, eine Dienstleistungsqualität von 150 % und Lebenszykluskosten von 50 % erklärte Ziele der Initiative Shift2Rail.

Ein Programm innerhalb von Shift2Rail beschäftigt sich mit der nächsten Generation Systemen zur Steuerung und Überwachung von Schienenfahrzeugen. Für dieses sogenannte *Train Control and Monitoring System* (TCMS) wird – wie für etwaige JBRs – ein Kommunikationsmedium für die Datenkommunikation zwischen den Subsystemen benötigt. Wie auch in bestehenden Fahrzeugen, soll dafür ein Fahrzeugbus zum Einsatz kommen. Die Neuerung hingegen ist, dass dieser Fahrzeugbus auf offenen Standards basieren und einheitlich für sämtliche (kritische und nichtkritische) Systeme auf dem Fahrzeug sein soll. Im Konkreten schlägt Shift2Rail für dieses sogenannte *Train Communication Network* (TCN) den Einsatz von Protokollen für TSN vor. TSN umfasst eine Reihe von Standards für Netzwerkprotokolle um eine hohe Verlässlichkeit (vorhersagbares Zeitverhalten, Verfügbarkeit) von Ethernet-Verbindungen zu ermöglichen.

Um die Zukunftsfähigkeit der Prototypen mit Bezug diese künftigen fahrzeugseitigen Technologien zu erproben, wurde im Kontext von *RailChain* ein Testaufbau für TSN realisiert. Für diesen Testaufbau wurde zunächst ein Netzwerk mit TSN zwischen zwei Industrie-Computer vom Typ Hewlett Packard Enterprise (HPE)

Edgeline EL20 hergestellt. Anders als übliche Systeme bzw. Netzwerkkarten, sind die EL20-Systeme mit Netzwerkkarten ausgestattet, mit denen Kommunikation via TSN möglich ist. Diese Netzwerkkarten unterstützen die für grundlegende Kommunikation via TSN benötigen Netzwerkprotokolle IEEE 802.1AS (Timing and Synchronization, auch Generalized Precision Time Protocol – gPTP) und IEEE 802.1Qav [39] (Forwarding and Queuing Enhancements for Time-Sensitive Streams). Konkret handelt es sich um Netzwerkkarten der i210-Serie von Intel.

6.2.2 Automatische Konfiguration der Testgeräte

Die Konfiguration auf Seiten des Betriebssystems wurde automatisiert und der wiederverwendbare Quelltext der Automatisierung öffentlich bereitgestellt [13]. Die Automatisierung erfolgte mit Hilfe des quelloffenen Automatisierungswerkzeugs Ansible[3] und umfasst

- die Prüfung der Kompatibilität zu TSN der Netzwerkkarten,
- die Konfiguration der für TSN erforderlichen virtuellen LANs (VLANs),
- die IP-Konfiguration der Netzwerkschnittstellen,
- letztendlich die Einrichtung der eigentlich für TSN benötigten Zeit-Synchronisation zwischen den Netzwerkkarten
- und optional die Einrichtung der Synchronisation der Systemuhren aller beteiligten Computer.

6.2.3 Co-Simulation mit *Marvis*

Die physische und betriebssystemseitige Installation wurde daraufhin zusammen mit einem unserer Prototypen im Open-Source-Testbed *Marvis* integriert. *Marvis* wurde und wird als offene und flexible Testplattform für die Evaluierung von verteilten Anwendungen mittels hybriden und co-simulierten Experimenten entwickelt. Komponenten des untersuchten Systems können also für eine hohen Grad an Skalierbarkeit in Software und für ein hohen Grad an Repräsentativität in Hardware in Versuche eingebunden werden (daher „hybrid“). Darüber hinaus könnten Teile der Systemumgebung mit domänenspezifischen Simulatoren nachgestellt werden (daher „Co-Simulation“). So wird in *Marvis* beispielsweise das Netzwerk zwischen Rechenknoten üblicherweise mit Hilfe des Netzwerksimulators *ns-3*[4] simuliert. Diese Netzwerksimulation ermöglicht es, verschiedene Netzwerktechnologien gegeneinander auszutauschen, zu erproben und Netzwerkverbindungen zu manipulieren (Fehlerinjektion).

Der hybride, co-simulierte Charakter von *Marvis* erlaubt es, realistische Szenarien im Sinne von *RailChain* nachzustellen. Konkret bedeutet dies, dass mindestens vier

[3] `https://www.ansible.com/`, besucht am 23.12.2022
[4] `https://www.nsnam.org/`, besucht am 23.12.2022

Rechenknoten für die Ausführung einer unserer Prototypen in *Marvis* bereitgestellt und vernetzt werden. Drei der vier Rechenknoten wurden virtualisiert umgesetzt, einer als separater physischer Computer. Zwischen dem Computer, der die virtualisierten Rechenknoten ausführt, und dem separaten Computer bestand das oben beschriebene Netzwerk für TSN. Zusätzlich zu diesem darunterliegenden Netzwerk wurde das Netzwerk aller vier Knoten in *ns-3* simuliert.

Eine weitere Standardisierung sieht die Middleware-Schicht *Train Real-Time Data Protocol* (TRDP) aufbauend auf TSN vor. TRDP soll im Zugumfeld gängige Kommunikationsmuster unterstützen, konkret den Austausch von Prozessdaten (z. B. Sensordaten) und Nachrichten. Im Hinblick auf die Zukunftssicherheit, kam im Projekt die Frage nach Vorteilen, Nachteilen und Aufwand der Nutzung von TRDP auf. Abseits unseres Prototyps im Speziellen, wird diese Frage vermutlich auch für bestehende Anwendungen im Allgemeinen zukünftig stellen.

6.3 Implementierung: Adaptierung der Implementierung

Prellblock ist eine Rust implementierter verteilter Ledger auf Basis von Punkt-zu-Punkt-Verbindungen. Diese Verbindungen sind mit TLS gesichert, die dazu nötigen Zertifikate werden vorab auf die Knoten verteilt, sodass die Kommunikationspartner sich gegenseitig authentifizieren können und die Kommunikation verschlüsselt abläuft.

Für das TRDP existiert eine Open-Source-Referenzimplementierung in der Programmiersprache C [14]. Um sie aus der Programmiersprache *Rust* nutzbar zu machen, musste ein *Wrapper* entwickelt werden. Zusätzlich mussten alle Schnittstellen zur Tranportschicht TCP/TLS in der Software gekapselt werden. Dazu zählt auch die Parametrisierung und Konfiguration, wie Zertifikate, Kennwörter, IP-Adressen der Blockchain-Knoten und Portnummern. Schließlich wurde dieselbe Schnittstelle durch eine neu geschaffene Komponente für TRDP implementiert, die sich die genannte Bibliothek für TRDP zunutze macht.

TRDP bietet zwei Kommunikationsmodi: Ein Prozessdateninterface, abgebildet über einen zyklischen, garantierten Zeitanteil auf dem Ethernet-Bus, sowie ein Messagedateninterface ohne Zeitgarantien. Für bessere Vorhersagbarkeit wurde für die *Prellblock*-Portierung das zyklische Prozessdateninterface gewählt. Auf Protokollebene wird die Kommunikation zwischen zwei oder mehreren Kommunikationspartnern über sogenannte *Topics* geregelt, ähnlich MQTT. Die *Prellblock*-Schnittstelle für die Transportschicht hat demgegenüber eine einfache Socket-Semantik, sodass die Transportkomponente von TRDP intern die Auswahl und Zuteilung von Topics vornimmt.

Es ist weiterhin möglich, TRDP auf „klassischem“ Ethernet ohne Zeitgarantien zu betreiben, sowie auf TSN (im Produktionsbetrieb). Im TCP/IP-Stack setzt TRDP auf UDP auf.

Im Testbed *Marvis* wurde die Implementierung auf korrekte Funktion geprüft und das tatsächliche Zeitverhalten auf einer TSN-tauglichen Hardware überprüft (Hardware-in-the-Loop-Tests).

6.4 Ergebnisse und Erkenntnisse

Für die experimentelle Evaluation der beschriebenen Implementierungen wurden der Durchsatz, die Latenz bis zum Konsens, sowie deren Schwankung (engl. *Jitter*) als Metriken in verschiedenen Szenarien erfasst.

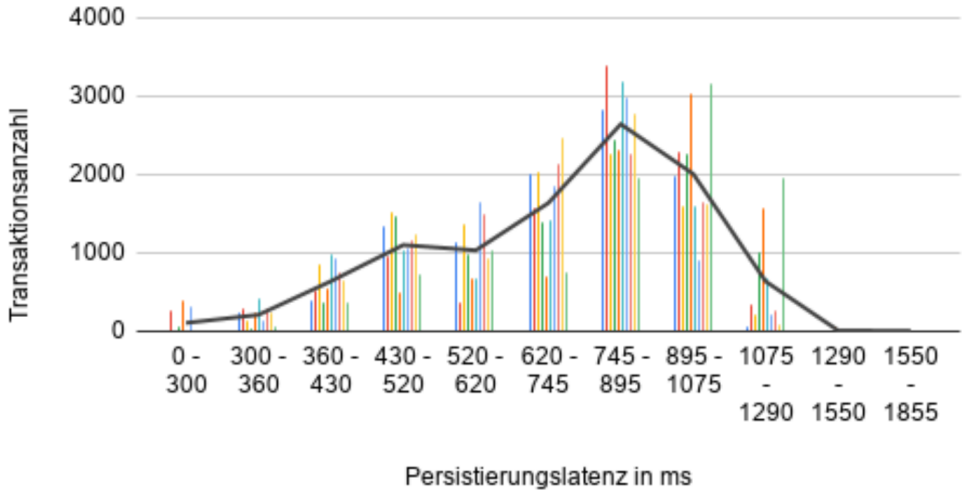

Abbildung 6.1: Histogramm der Latenzen bis zur konsensbasierten Speicherung der Daten. Jeder farbige Balken steht für eine der 10 Wiederholungen. [1]

Wie in Abbildung 6.1 angedeutet, konnte für die *Prellblock*-Implementierung gezeigt werden, dass die von der Spezifikation geforderte Latenz zur gerichtsfesten Speicherung in den meisten Fällen eingehalten werden kann. Anhand der Tests im Labor kann somit das vorgeschlagene Konzept als grundsätzlich geeignet beurteilt werden.

Des Weiteren wurde die *Prellblock*-Implementierung mit den etablieren *Hyperledger*-Plattformen *Fabric* und *Sawtooth* verglichen. In Abbildung 6.2 ist unter anderem zu erkennen, dass der Durchsatz von *Prellblock* in allen untersuchten Szenarien mindestens eine Größenordnung höher war als der der anderen beiden Implementierungen. Die Reduktion auf die für den Anwendungsfall relevanten Funktionalitäten, sowie die Ausnutzung von domänen- und anwendungsspezifischen Annahmen ermöglicht diese höhere Leistungsfähigkeit im Vergleich zu generischen Implementierungen von DLTs. Diese Tatsachen erlauben es zusätzlich, den zunächst intuitiv herausfordernd wirkenden Einsatz von DLTs für Echtzeitanwendungen wie *RailChain* als durchaus realisierbar zu bewerten.

Um die Fehlertoleranz des Software-Prototyps *Prellblock* zu untersuchen, kam die Methodik der Fehlerinjektion zum Tragen. Bei dieser Methodik werden angenommene Fehlerursachen im System herbeigeführt und es wird beobachtet, ob sich das System erwartungsgemäß verhält. Im Folgenden wird beispielhaft ein besonders interessantes Szenario und dessen Messwerte der Fehlerinjektionskampagne vorgestellt. Als Fehlerursache wurde dabei der Ausfall einer der vier Knoten

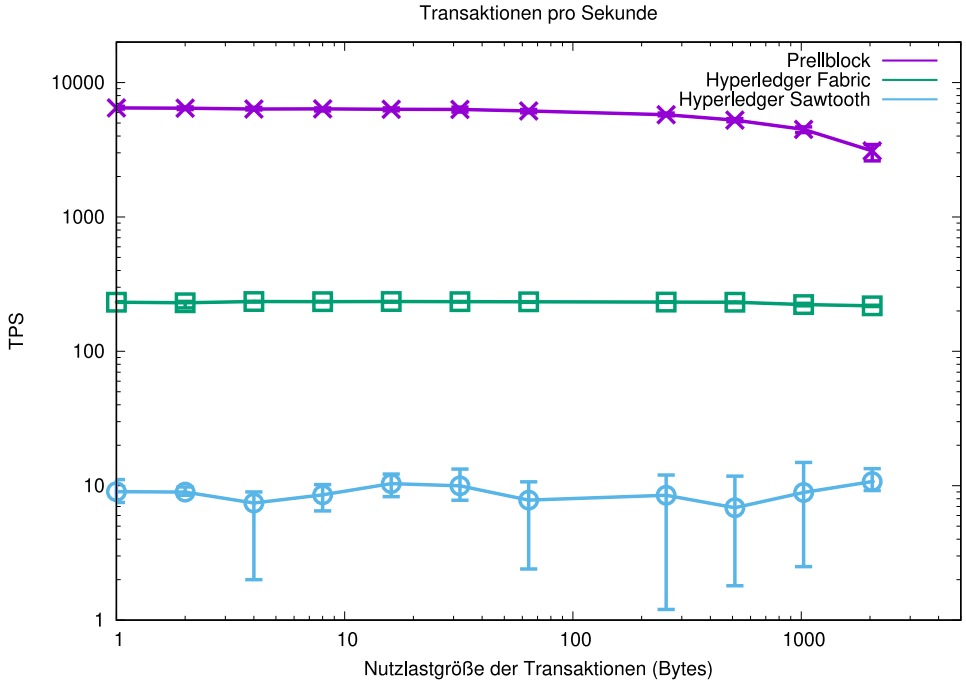

Abbildung 6.2: Transaktionen pro Sekunde (TPS) in Relation zur Nutzlastgröße. Es wird deutlich, dass das Ausnutzen von anwendungsspezifischen Annahmen in angepassten Implementierungen (hier *Prellblock*) im Vergleich zu generischen Implementierungen (hier *Fabric* und *Sawtooth*) eine höhere Leistung ermöglicht. [25]

angenommen (für die anderen Knoten erkennbar, engl. *crash fault*). Während das System also mit eingehenden Daten, also auch Transaktionen, konfrontiert ist, wird ein Knoten zum Absturz gebracht. Ähnlich zu Abbildung 6.1 wurden die Latenzen bis zur konsensbasierten Speicherung der Daten gemessen und als Gütekriterium betrachtet. Abbildung 6.3 zeigt die gemessenen Latenzen bis zur konsensbasierten Speicherung der Daten mit einem ausgefallenen *Prellblock*-Knoten.

Ein Großteil der Latenzen bewegt sich in einem ähnlichen Bereich wie die Latenzen der Messung ohne injizierte Fehler (siehe Abbildung 6.1). Auffällig ist jedoch, dass einige Transaktionen erst nach vergleichsweise sehr langem Verzug persistiert worden sind (14,5 Sekunden, 99. Perzentil). Dieser Fall ist – unter hoher Last des Systems durch weiterhin künstlich hohe eingehende Datenmengen – mit dem Ausfall des Anführers und der daraufhin erforderlichen Wahl eines neuen Anführers zu begründen (ähnlich zum *View Change* der ZugChain-Implementierung, siehe Unterabschnitt 4.4.1). Nichtsdestotrotz gehen auch in diesem Fall keine Daten verloren, da die Implementierung von *Prellblock* das Konzept der gestaffelten Konsistenz (engl. *tiered consistency*) einführt. Dabei werden die zu persistierenden Daten pro Knoten zunächst in einen lokalen, nichtflüchtigen Speicher (z. B. Datenbank) geschrieben. Sollte in Fehlerfällen kein Konsens erreicht werden können (z. B. auch bei Ausfall des Netzwerks), werden so trotzdem alle Daten redundant persistiert. Liegt kein Fehler vor bzw. erholt sich das System von zuvor aufgetretenen Fehlern, werden die Daten der Konsensfindung zugeführt und in der Folge für gerichtsfeste Korrektheit und Unveränderlichkeit in die Blockchain überführt.

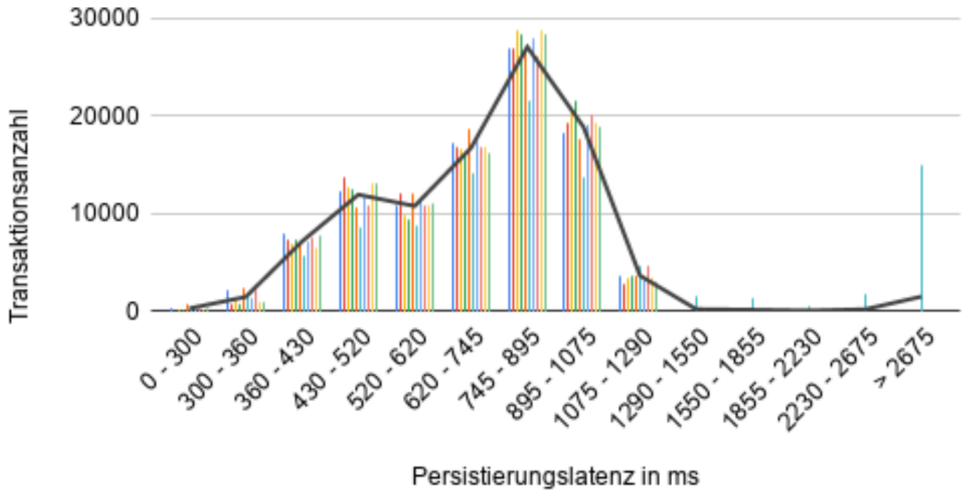

Abbildung 6.3: Histogramm der Latenzen bis zur konsensbasierten Speicherung der Daten während des Ausfalls eins *Prellblock*-Knotens. Größtenteils wurden Latenzen ähnlich zu den Messungen am fehlerfreien Systems beobachtet. Die wenigen erhöhten Latenzen (türkis dargestellt) sind durch den Ausfall des Anführers zu begründen, der während der hohen Testlasten auftrat. [1]

6.5 Infrastruktur für die Demonstration

Nicht nur für die Demonstration, auch für die Entwicklung und das Testen der Implementierungen musste der Betrieb von IT-Infrastruktur an Bord des Versuchsträgers ermöglicht werden. Dabei stellten sich nicht nur rechtliche und räumliche Fragen als herausfordernd heraus (siehe Abschnitt 7.2), sondern auch scheinbar banale Aspekte wie die Bereitstellung eines lokalen Netzwerkzugangs sowie die Stromversorgung der Computer- und Netzwerkgeräte.

Da im Fahrgastraum des Versuchsträgers nur temporär Netzwerkkabel verlegt werden konnten, wurde im Zuge von *RailChain* zusätzlich zum lokalen, kabelgebundenen Zugang zum Testnetzwerk ein drahtloses Netzwerk für die Entwicklung und Tests an Bord bereitgestellt.

Die Stromversorgung elektrischer Verbraucher auf dem Versuchsträger erfolgt wie üblich über mehrere Bordnetze. Neben verschiedenen elektrischen Eigenschaften haben diese Bordnetze oft eine Einschalt- und Ausschaltreihenfolge, sowie unterschiedliche Wichtigkeiten für den Betrieb des Zuges. Wie das Projekt in den ersten Tests feststellen musste, kann nicht bei allen Bordnetzen davon ausgegangen werden, dass während der gesamten Dauer einer Testfahrt ununterbrochen funktionieren. Während die an den MVB angeschlossenen Geräte während der Testfahrten ununterbrochen mit Strom versorgt wurden, wurde die Stromversorgung von Computern und Netzwerkgeräten, die die Entwicklung und Tests unterstützten, teilweise unterbrochen. Zur Umgehung dieses Umstandes wurde eine unterbrechungsfreie Stromversorgung für Entwicklungs- und Demonstrationscomputer (220 VAC), sowie für Netzwerk-Hardware (12 VDC) konzipiert und umgesetzt. Maßgebliche Aspekte dabei waren Kapazitäten (Strom/A, Energie/Ah).

7 Demonstrationsbetrieb

Der Demonstrationsbetrieb ist seit Anfang an ein fester Bestandteil des *RailChain*-Projekts. Zum einen kann damit das Zusammenspiel des Gesamtsystems in praktischer Anwendung zu erprobt werden. Nicht zu vernachlässigen sind dabei auch die Herausforderungen der Einbettung in bestehende technische und organisatorische Systeme, sowie die Interaktion mit diesen umgebenen Systemen. Zum anderen kann das Projekt so mit allen drei Use Cases bei öffentlichen Präsentationen dem Fachpublikum und interessierten Laien zugänglich gemacht werden.

In diesem Kapitel werden die Aktivitäten, Ergebnisse und Erkenntnisse der verschiedenen Demonstrationen des *RailChain*-Projekts dargestellt. Neben der Interaktion mit dem *IDunion*-Netzwerk für Use Case 1, liegt das Augenmerk auch auf dem Einbau eines Prototypen in den Versuchsträger *advanced TrainLab*s der Deutschen Bahn (ICE-TD 605 017) für Use Case 2 und Use Case 3.

7.1 *RailChain Asset Identity* Use Case 1

Das Prinzip der *Asset Identity* funktioniert über registrierte Teilnehmer auf dem *IDunion*-Netzwerk:

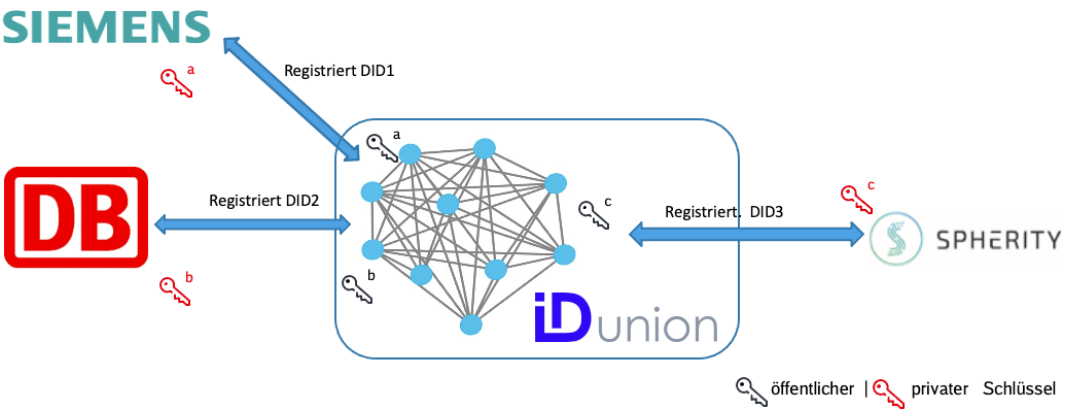

Abbildung 7.1: Teilnehmer registrieren sich im *IDunion*-Netzwerk

Über die quelloffen verfügbare Software kann jeder Teilnehmer zu einem anderen Teilnehmer Verbindungen aufbauen und signierte Daten austauschen, die mit Hilfe des *IDunion*-Netzwerkes verifizierbar sind:

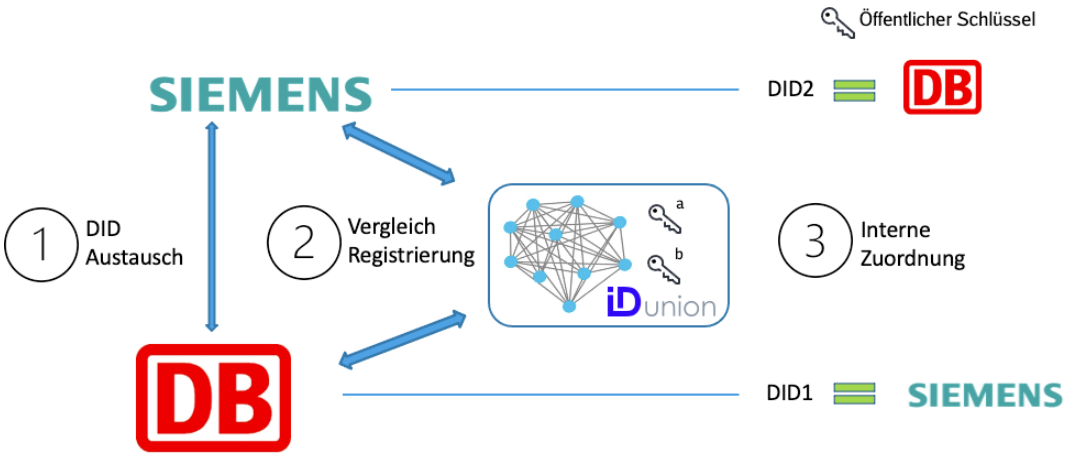

Abbildung 7.2: Kontaktaufnahme zwischen Kommunikationspartnern und Verifizierung über *IDunion*

Mit dem Aufbau des Verbindungskanals der Partner ist eine vertrauenswürdiger Kommunikationskanal zwischen diesen geschaffen und vom Partner ausgestellte Zertifikate (sog. *Verifiable Credentials* [5]) können zugeordnet und sicher verifiziert oder auch zurückgezogen werden.

Ein digitales Asset, das z. B. von der Siemens AG hergestellt wird, bekommt von der Siemens AG ein *Verifiable Credential* (VC) mit typischen Merkmalen, erfüllten Standards und eindeutigen Daten (z. B. Seriennummer) ausgestellt:

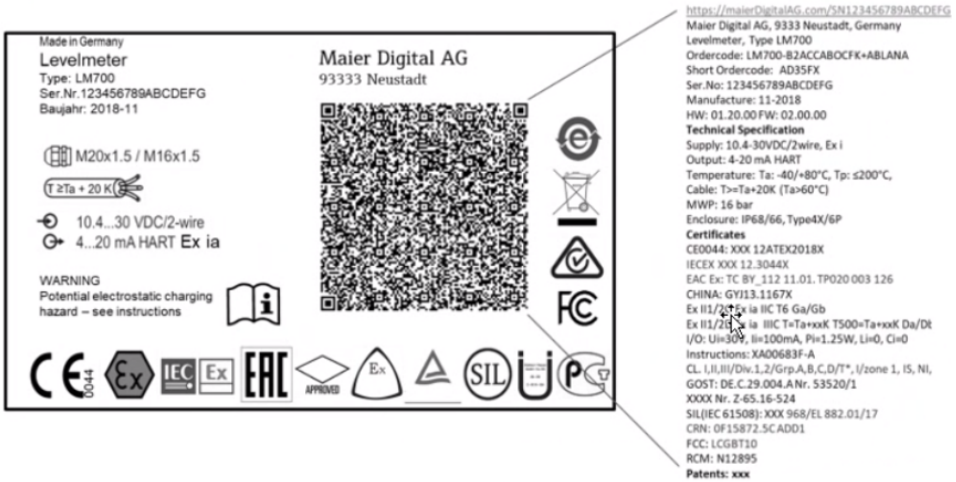

Abbildung 7.3: Beispiel für ein digitales Typenschild

Zudem gibt Siemens dem Asset ein weiteres Zertifikat mit, das bezeugt, dass das Asset Siemens gehört (Besitzer-VC).

Durch einen voll automatisierbaren Prozess ist es nun möglich den Verkauf des Assets an die Deutsche Bahn sowohl in externen Systemen als auch dem Asset selbst durch eine Änderung des Besitzer-VCs „bewusst“ zu machen. Selbstverständlich muss der Vorbesitzer diesem Vorgang zustimmen:

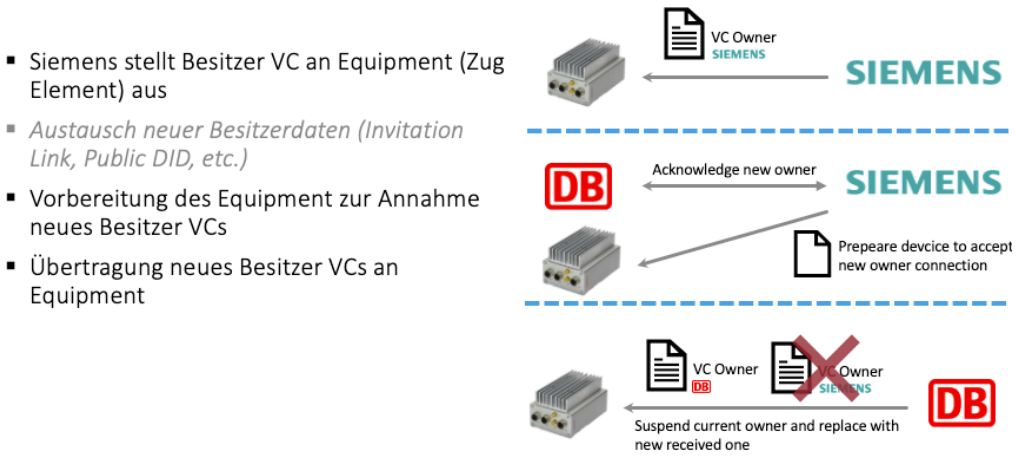

Abbildung 7.4: Besitzerwechsel *Asset Identity*

Mit dem Wissen wer sein Besitzer ist, lassen sich über die Identitäten der beteiligten Firmen sehr interessante Szenarien aufbauen.

Sollte die DB AG einem Wartungspartner den Auftrag erteilen, die Wartung für eine Baureihe zu übernehmen, so müssen folgende Voraussetzungen erfüllt sein:

- Der Hersteller (Siemens) hat dem Wartungspartner ein VC ausgestellt, dass ihn für die Wartung einer Baureihe qualifiziert.
- Die DB AG als Besitzer hat den Wartungsauftrag per VC an den Wartungspartner vergeben. Dabei kann eine ganze Reihe von Assets oder nur ein bestimmtes Gerät.

Im folgenden Ablauf wird der Wartungspartner (hier Spherity) eine Verbindung mit dem Gerät aufbauen. Ob das Asset diese Verbindung annimmt, hängt vom Besitzer-VC mit der Berechtigung ab. Ob der Wartungspartner ein VC vom Hersteller hat, das ihn zur Wartung berechtigt, ist eine weitere Prüfung (siehe Abbildung 7.5).

Der Wartungspartner kann die Verbindung nach Durchführung der Wartung auch nutzen, um dem Asset ein Wartungs-VC auszustellen, dass die ordnungsgemäße Durchführung nachweisbar macht (siehe Abbildung 7.6).

Diese schematisch dargestellten Abläufe sind zwischen den Partnern DB, Siemens und Spherity an realen Assets demonstriert worden:

Leider ließen sich viele angedachte Szenarien im Use Case 1 nicht umsetzen, da bei der Spezifikation von einer Einbaufähigkeit in ein reales Zugumfeld ausgegan-

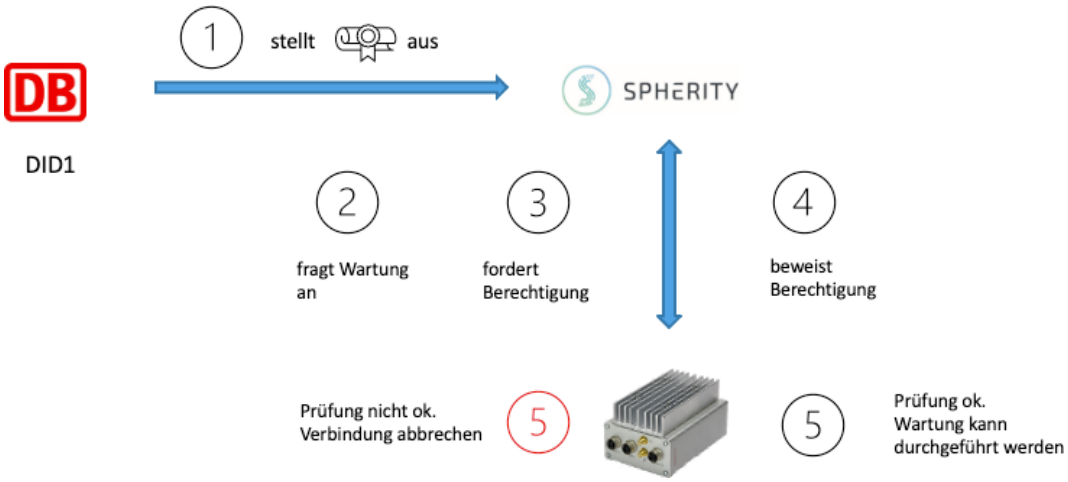

Abbildung 7.5: Wartungspartner weist sich aus

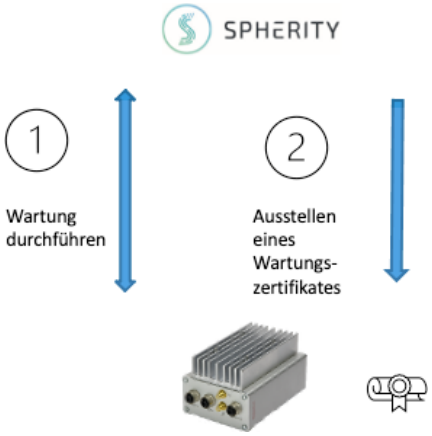

Abbildung 7.6: Ausstellen eines Wartungs-VC

Abbildung 7.7: Demonstration *Asset Identity*

gen worden ist. Aus Gründen der Rückwirkungsfreiheit des Versuchsaufbaus ist keinerlei Datenaustausch zwischen den Systemen möglich gewesen.

7.2 Umfeld und Aufbau Demonstration Use Case 2 & Use Case 3

Neben der Demonstrationsmöglichkeit des Laboraufbaus auf Messeständen hatte das Projekt von Beginn an das Ziel die praktische Umsetzung in einem realen Zugumfeld zu demonstrieren.

Die wesentliche Hürde dabei ist die Betriebszulassung des Zuges nicht zu gefährden. Eine wesentliche Voraussetzung dafür ist der Zugang zu einem Zug und der nachweislich rückwirkungsfreie Aufbau der Komponenten.

Ein weiterer Aspekt ist die Demonstration der Herstellerunabhängigkeit der Systemkomponenten. Ausgehend vom Versuchsaufbau des Laborsystems konnten Rahmenparamter für die Dimensionierung der Hardware im Zugdemonstrator abgeleitet werden. Die ausgewählten Komponenten sind für den Zugbetrieb zertifizierte Elemente, die vom offenen Markt bezogen werden konnten.

7.2.1 Versuchsträger *advanced TrainLab*

Das *advanced TrainLab* der Deutschen Bahn ist ein Versuchsträger und Demonstrationszug, um neue und innovative Technologien im realen Schienenverkehr testen zu können. Nach Vorstellung der Projektidee hat sich das *advanced TrainLab* dafür entschieden *RailChain* die Chance einer Demonstration zu eröffnen.

Der Zugang zu den MVB-Daten sollte im vorgesehenen Design an vier verschiedenen Stellen erfolgen. Für einen rückwirkungsfreien Aufbau war es nötig, eine Systemkomponente der DB Systemtechnik zwischen den *Multifunction Vehicle Bus* (MVB) und den vier Blockchain-Knoten zu schalten.

Eine weitere Voraussetzung für die Machbarkeit ist ein Systembus, der die mindestens vier Rechnerknoten miteinander verbindet, um den Konsens zwischen den Knoten zu verhandeln. Der MVB im *advanced TrainLab* ist dafür nicht geeignet, d. h. für den Demonstrationsaufbau war ein eigenes lokales Netzwerk erforderlich.

In mehreren Vor-Ort-Terminen stand die Suche nach einem geeigneten Einbauort im Vordergrund. Auch Aufgrund der physischen Gegebenheiten im *advanced TrainLab*, konnten die Knoten des Demonstrators nicht über die Länge des Zuges verteilt werden. Bezogen auf die Ausfallsicherheit bei physischen Einwirkungen, ist dies eine Einschränkung gegenüber einer für den Regelbetrieb gedachten Konstellation.

Es konnten keine vier unterschiedlichen Zugangsstellen zum MVB ausgemacht werden, die noch genügend Platz für einen verteilten Systemaufbau zugelassen hätten. Daraus folgte, dass die Informationen des MVB nur an einer Stelle vom Zug entnommen und über das lokale Netzwerk auf die vier Knoten verteilt werden.

Diese Einschränkungen sind für eine Machbarkeitsdemonstration in einem realen Zug als vernachlässigbar einzustufen. Die Konstellation ermöglicht immerhin

einen kompakten Einbau aller Komponenten im Zugbegleiterabteil für eine gute Präsentierbarkeit der Projektergebnisse.

7.2.2 Versuchsaufbau *RailChain* auf dem Versuchsträger

Der komplette Versuchsaufbau des Zugdemonstrators ist in folgendem Bild dargestellt:

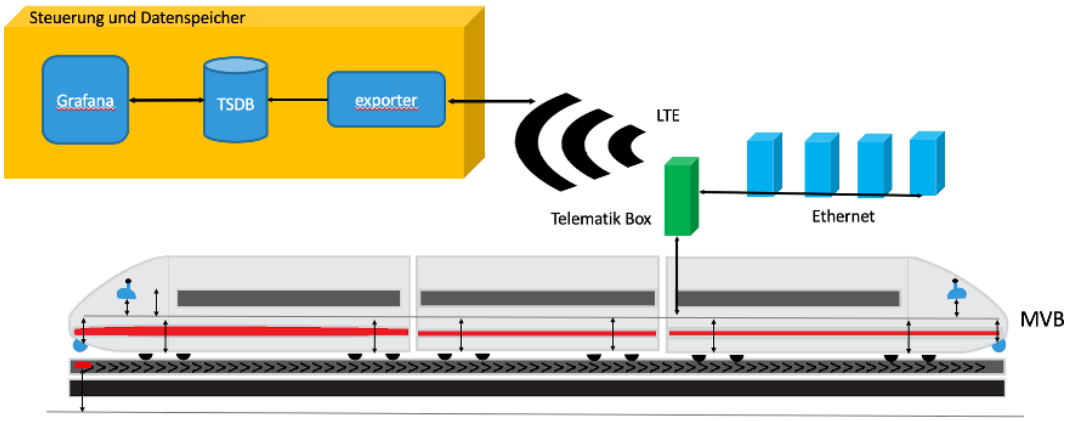

Abbildung 7.8: Schematischer Aufbau des Demonstrators auf dem *advanced TrainLab*

Ausgehend vom MVB, dessen Daten von der Telematik-Box der DB Systemtechnik aufgegriffen, gefiltert und über das lokale Ethernet allen vier Knoten gleichzeitig zur Verfügung gestellt werden, verarbeiten die vier Rechnerknoten diese und gleichen sie über das Konsensprotokoll untereinander ab.

Ein weiterer Vorteil der Telematik-Box der DB Systemtechnik ist die Erreichbarkeit über eine LTE-Verbindung über eine Dachantenne des Zuges. Das eröffnet die direkte Fernüberwachung des Systems, die Nachverarbeitung und die Aufbereitung der Daten via Internet. Zudem hat die Möglichkeit des Fernzugriffs dem Projekt außerordentlich über die Pandemiezeit geholfen.

Die folgende Abbildung 7.9 zeigt den Versuchsaufbau im Zug:

Die Zugposition wird über eine GPS-Antenne am Dach des *advanced TrainLab* auf einen der Versuchscomputer erfasst und kann für die weitere Auswertung zur Verfügung gestellt werden.

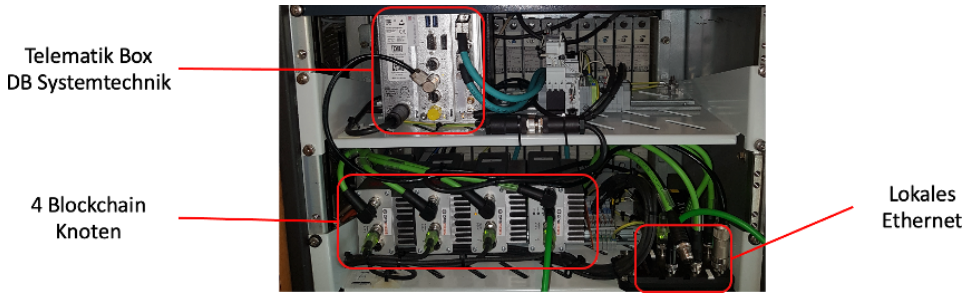

Abbildung 7.9: Aufbau des Demonstrators im *advanced TrainLab*

7.3 Ergebnisse des Demonstrationsaufbaus

Aus dem Laborumfeld wusste das Projekt um die Leistungsfähigkeit des Systems und der Software. Allerdings war es für das Projekt dennoch sehr spannend zu sehen, wie sich das System im realen Zug verhält und wie es mit den realen Zugdaten umgehen kann.

Auf dem MVB des *advanced TrainLab*s fallen insgesamt 12.000 Signale (Messwerte oder Daten) an. Die im *advanced TrainLab* verbauten *Juridical Recording Unit* (JRU) sind im Originalzustand auf 32 Werte konfiguriert. Dies entspricht dem Entwurf der Entstehung der Baureihe VT605 aus den 1990er Jahren. Moderne Konfigurationen einer JRU sind auf ca. 70 Werte ausgelegt.

In der Projektwoche vom 05.-09.07.21 wurde das System ausgiebig getestet und es konnte ein stabiler Betrieb bei mehr als 800 Signalen nachgewiesen werden. Eine weitere Optimierung ist sicher möglich. Für die weiteren Tests ist allerdings eine Anzahl von 400 sinnvollen Signalen ausgewählt worden. Für den JRU-Betrieb ist das weit mehr als ausreichend, für andere Zwecke der Datenaufzeichnung und späteren Auswertung wird diese Leistungsfähigkeit als ausreichend eingeschätzt. Welche Werte aufbereitet werden, kann an der Telematik-Box konfiguriert und den Umständen entsprechend angepasst werden.

In Abbildung 7.10 und Abbildung 7.11 werden Daten einer Testfahrt am 18.11.21 von Potsdam nach Stendal dargestellt. Die in Abbildung 7.11 dargestellten MVB-Daten sind ein Auszug aus 400 Zugdaten, die während der Fahrt im *Juridical Blockchain Recorder* (JBR) aufgezeichnet, über das Netz im verschlüsselten Format übertragen und dann in der Cloud für die Auswertung und Darstellung verarbeitet wurden. Es ist ein reales Fahrverhalten mit Beschleunigungen, Bremsvorgängen und Zwischenstopps zu erkennen.

Sehr interessant ist das Zeitverhalten des Konsensmechanismus während dieser Fahrten. Die Anforderung von einer Speicherung innerhalb 1 Sekunde ist in der JRU-Norm gefordert. Eine lokale Zwischenspeicherung der Daten erfolgt auf jedem Knoten separat in Mikrosekunden im internen Flash-Speicher der Computerknoten. Für die JBR-Funktion ist allerdings entscheidend, wann auf jedem Knoten der Konsenszustand (Bestätigung, dass alle anderen Knoten den gleichen Datenbestand haben) nach Dateneingang erreicht ist:

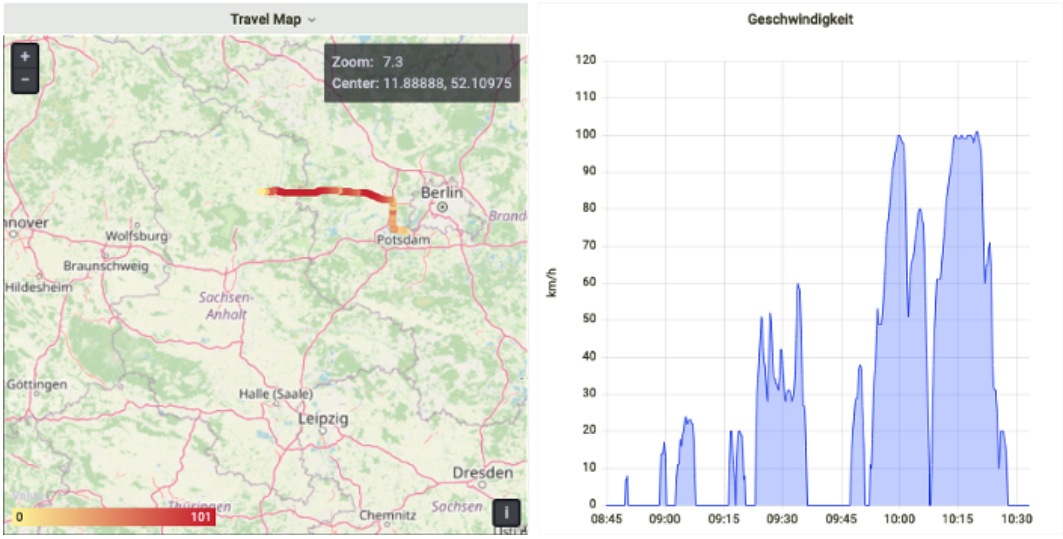

Abbildung 7.10: Geodaten der Versuchsfahrt am 18.11.21.

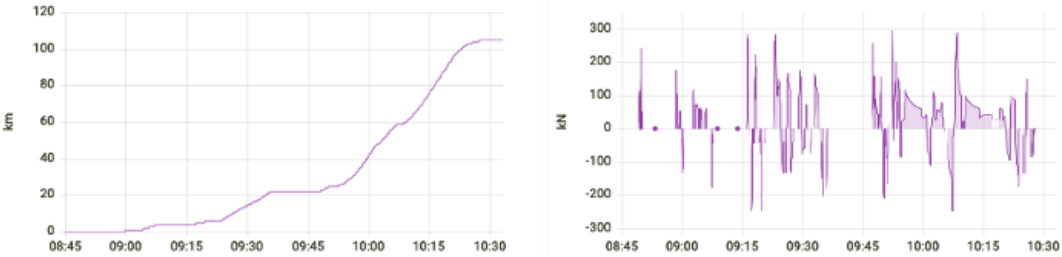

Abbildung 7.11: MVB-Daten: Tageskilometer und Beschleunigungs- und Bremskräfte

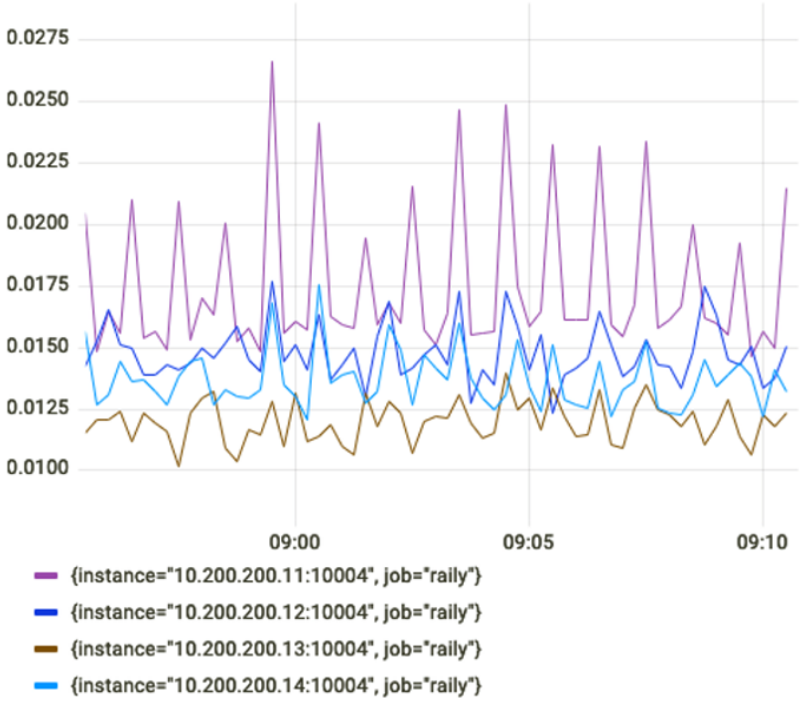

Abbildung 7.12: Latenz zwischen Dateneingang und Konsens in ms

Die Messergebnisse zeigen durchgängig ein Erreichen des Konsens nach 0,03s in manchen Fällen erst nach 0,06 Sekunden. Jede der oben dargestellten Linien zeigt die Latenz auf einem der vier Knoten. Ein Knoten ist immer der letzte (hier die grüne Linie), das ist der Masternoten. Dieses Verhalten ist absolut normal und im Rahmen der Erwartungen, da der Masterknoten als letzter von allen anderen die Bestätigung der Gleichheit der Prüfsummen erhält. In manchen Diagrammen sieht man auch den Wechsel des Masterknotens an dem Wechsel der Farben der Linien.

Mit diesen Darstellungen kann deutlich demonstriert werden, dass die Konsensfindung nach 0,06 Sekunden zwischen den Knoten abgeschlossen ist und damit die wesentliche Anforderung an die Geschwindigkeit des Abgleichs bei deutlich mehr Signalwerten als nötig erfüllt werden kann.

7.4 Messestand *Demo Day* der *Digital Rail Convention* 2021

Auf dem *Demo Day* der *Digital Rail Convention* 2021 in Scheibenberg ist *RailChain* über einen Messestand vertreten. In Abbildung 7.13 und Abbildung 7.14 sind das Testbed und die Messeaufbauten zu sehen.

Abbildung 7.13: Projekt *RailChain* im regen Austausch mit Fachpublikum

Weiterhin ist der Einbau in den *advanced TrainLab* live auf Demofahrten zwischen Markersbach und Schlettau demonstriert worden (Abbildung 7.15).

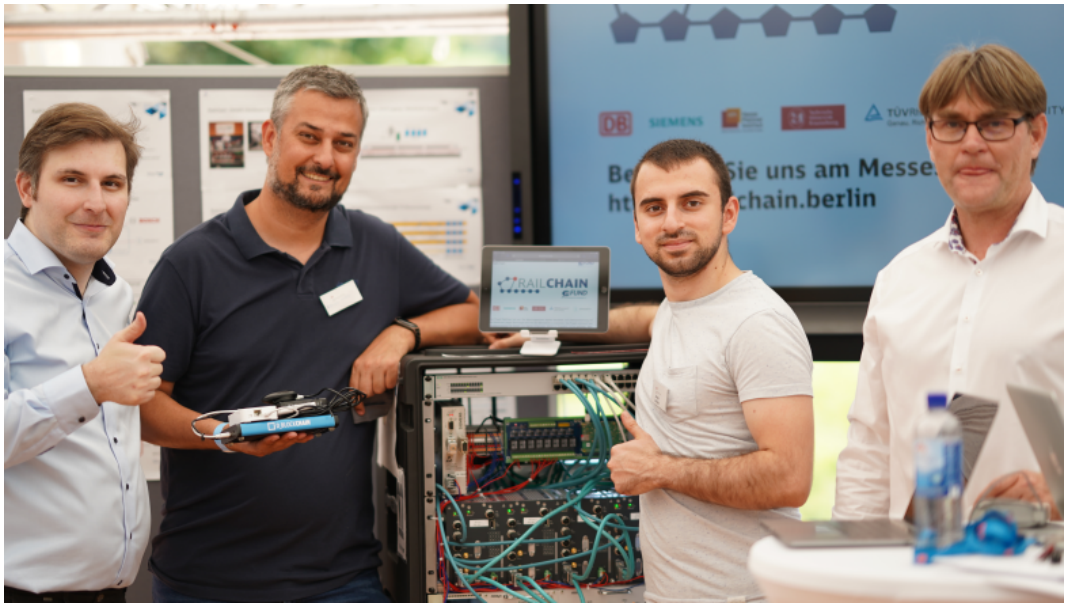

Abbildung 7.14: Projekt *RailChain* inkl. Testbed bei dem *Demo Day* der *Digital Rail Convention* 2021 in Scheibenberg

Abbildung 7.15: Der Testaufbau im *advanced TrainLab*

Abbildung 7.16: Das *advanced TrainLab* zwischen dem Fachpublikum beim *Demo Day* der *Digital Rail Convention* 2021 in Scheibenberg

7.5 Weitere Präsentationen

Die Fragestellungen und Erkenntnisse aus dem Projekt wurden im akademischen Umfeld nicht nur in der Forschung bearbeitet, sondern sind auch in die Lehre eingegangen. Zusätzlich wurden die Themen in und um *RailChain* von allen Partnern bei öffentlichkeitswirksamen Veranstaltungen präsentiert, kommuniziert und diskutiert:

- Bachelorprojekt „IoT, Sensorik, Nachvollziehbarkeit – die Software-Blackbox“ 2019/2020
- Bachelorprojekt „Die IoT-Middleware im Zug: Vernetzte Sensorik für zustandsorientierte Instandhaltung“ 2020/2021
- Projektvorstellung und Partnerangebot *Main Incubator* (*IDunion*, vormals *Lissi*) 16.09.20
- mFund Fachaustausch Blockchain 14.10.2020
- Vorlesung „Eingebettete Betriebssysteme“ 2020/2021
- Bachelor Software-Entwicklungspraktikum „Data Logging on Trains Using Blockchains“ 2020
- Digital Rail Summer School 2020
- Vorlesung „Eingebettete Betriebssysteme“ 2021/2022

- Digital Rail Summer School 2021
- DB Meetup DSD Community 28.10.21
- Digital Rail Demo Day 2021
- Digital Rail Summer School 2022
- Praktikum/Wettbewerb Verteilte Systeme: „Performance Attacks on Byzantine Fault-Tolerant Systems“ 2022
- Demofahrten aTL Berlin 2022
- Blockchain Autumn School 2022 BCCM
- InnoTrans Berlin 2022
- Tag der deutschen Einheit Stand Land Berlin in Erfurt 2022

Erfolgreiche Demonstration des Projektes auf der Abschlussdemonstration in der Woche vom 29.08.–02.09.2022 in Anwesenheit von Politik, Eisenbahnindustrie, DB AG und Kinder aus dem Kindergarten.

Auf der InnoTrans wurden vielfältige Gespräche geführt und weitere Möglichkeiten einer Verwertung thematisiert.

Zum 32. Tag der deutschen Einheit erhielt das Projekt viel Aufmerksamkeit durch Politik und Publikum.

Durch diese vielschichtige Verbreitung und Kommunikation der Projektinhalte gewannen Lehre, Forschung, Industrie und Praxis Nähe zueinander. Der regelmäßige Austausch zwischen allen Beteiligten – insbesondere zwischen Forschenden, Studierenden, Industrie, Politik, Bürgerinnen und Bürger – wurde als Gewinn für alle Seiten wahrgenommen.

8 Standardisierung

Die Normung von Digitaltechnologien für Eisenbahnen findet prinzipiell auf drei unterschiedlichen Ebenen statt:

- Weltweit bei der *International Electrotechnical Commission* (IEC)
- Europaweit bei der *Commisision Europeenne de Normalisation Electrotechnique* (CENELEC)
- National bei der Deutschen Kommission für Elektrotechnik (DKE)

Dabei wird nach dem Prinzip vorgegangen, möglichst weiträumig und umfassend zu normen. Das sog. *Frankfurt Agreement* räumt, falls möglich, weltweiter Normung Vorrang vor regionaler oder nationaler Normung ein. So wird z. B. in IEC 62625 [35] das „On-board Data Recording System" prinzipiell standardisiert. Bisher hat es allerdings im Bereich der Eisenbahnen keine spezifischen Normungsaktivitäten zu Themen der *Distributed-Ledger*-Technologien (DLTs) gegeben.

Wie im Bereich der Informationstechnologie üblich, werden viele allgemeine Themen bei der International Standards Organisation (ISO) genormt, so auch im Bereich DLT im ISO/TC 307 „Blockchain and Distributed Ledger Technologies". Allerdings blieb auch hier die Normung eher allgemein, z. B. zur Terminologie.

Für *RailChain* relevant sind hier insbesondere Veröffentlichungen zum Thema Use Cases, die allerdings in der Regel nicht den Status von Normen, sondern nur von technischen Spezifikationen haben:

- ISO 23257 [40]: Blockchain and distributed ledger technologies – Reference architecture, 2022
- ISO/TR 3242 [41]: Blockchain and distributed ledger technologies – Use cases
- DIN SPEC 3103 [10] Blockchain und Distributed Ledger Technologien in Anwendungsszenarien für Industrie 4.0, 2019

Verglichen mit den Projektergebnissen zeigt sich, dass die Passfähigkeit zu den Standards gut gegeben ist, wenn diese mehrheitlich während der Bearbeitung noch nicht ausgearbeitet waren. So zeigt sich z. B. eine gute Übereinstimmung mit der UML-basierten Beschreibung der Use Cases in DIN SPEC 3103 als auch mit den Referenzarchitekturen in ISO 23257. In der ISO 23257 werden insbesondere *Interoperability Interfaces*, *DLT data flows* sowie *System Views of Functional Components* thematisiert.

Im Sinne der o. g. Standards kann das *RailChain*-Konsensprotokoll auch als sog. *Smart Contract* aufgefasst werden: „Ein Smart Contract stellt ein Programm

dar, das in der Blockchain verteilt gespeichert wurde und durch einen beliebigen Systemteilnehmer ausgeführt werden kann. Die Ausführungsergebnisse werden wiederum auf der Blockchain gespeichert und allen Systemteilnehmern zugänglich gemacht". Das folgende Zitat aus der DIN SPEC 3103 mag die Passfähigkeit unterstreichen: „Hierzu ist eine Rahmenarchitektur erforderlich, die einerseits eine sichere, leichtgewichtige Anbindung der Sensoren an die Blockchain bzw. DLT sowie die *Smart Contracts* definiert. Andererseits müssen Struktur und Ablaufmechanismen der *Smart Contracts* selbst beschrieben werden, damit diese eine legale Akzeptanz erreichen können."

Allerdings bieten sich in der Eisenbahnnormung keine adäquaten Anknüpfungspunkte bei IEC oder CENELEC. Die Standardisierung der Referenzarchitekturen oder der Schnittstellen findet zumindest in Europa im Rahmen privatwirtschaftlich geprägter Initiativen wie OCORA (*Open CCS On-board Reference Architecture*) oder EULYNX statt bzw., wenn die Interoperabilität betroffen ist, in der TSI der ERA bzw. den dort referenzierten Dokumenten.

8.1 Standardisierung der FIS für den *Juridical Blockchain Recorder* (JBR)

Daher wurde im Rahmen von *RailChain* eine *Functional Interf ace Specification* (FIS) für den JBR (JBR) ausgearbeitet. Das Dokument gibt eine Übersicht über alle Schnittstellen, die für die Funktionalität des JBR von Nöten sind und wie die Kommunikation zwischen den Schnittstellen stattfindet, insbesondere im Hinblick auf die Reihenfolge von Ereignissen und Nachrichten. Hierzu wurden die Schnittstellen in drei verschiedene Kategorien eingeteilt:

1. Schnittstellen, die mit der Aufnahme juristisch relevanter Daten verbunden sind. Dabei hängt die genaue Definition von juristischen Daten vom Betreiber, der verwendeten Bus-Architektur und vom verwendeten Zugbeeinflussungssystem ab. Generell lassen sich juristische Daten in zwei Kategorien einteilen: Prozessdaten (Echtzeitdaten für Zugkontrolle und -überwachung, bspw. Geschwindigkeit, Position) und Messagedaten (auf Anfrage bereitgestellte Daten). Da der JBR-Ansatz nachrichtenagnostisch ist, ist eine genaue Angabe der Nachrichtenform nicht nötig. Im Rahmen von ERTMS/ETCS kann aber beispielsweise die FIS der *Juridical Recording Unit* (JRU) als Beispiel herangezogen werden. Diese geht von einer Ethernet-Bus-Architektur aus, wie auch im Demonstrator umgesetzt.

2. Schnittstellen, die bei der Kommunikation der zugseitigen Blockchain Knoten untereinander gebraucht werden. Hierzu zählt insbesondere sämtliche Kommunikation, die zur Konsensfindung und zur Detektion fehlerhafter Knoten benötigt wird.

3. Schnittstellen, die bei der Synchronisierung mit Datenzentren benötigt werden. Dies schließt die Datenübertragung und Identifikation von Datenzentren mit ein.

Für Punkt 2 und 3 wurden die genutzten Schnittstellen und Nachrichtenstrukturen innerhalb der FIS aufgeschlüsselt, für Punkt 1 ist dies aufgrund der durch Betreiber, Zugbeeinflussungssystem und Bus-Architektur bestehend Variabilität jedoch nicht sinnvoll und aufgrund der Agnostik des Ansatzes auch nicht nötig. Als Grundlage für die Aufschlüsselung der Punkte 2 und 3 wurde die konkrete Implementation von ZugChain genutzt. Die genaue Angabe der Nachrichtenform für die Kommunikation zwischen den Knoten und mit dem Datenzentrum ermöglicht es interessierten Parteien, alle Schnittstellen entsprechend ihrer Spezifikation anzusteuern und dem System alternative oder neue Funktionalitäten hinzuzufügen.

Eine zusätzliche Grundlage hierfür ist die herkömmliche *Functional Interface Specification* (FIS) *Juridical Recording* (Subset 027) (*European Railway Agency*), die eine Nutzung im Rahmen von ETCS (*European Train Control System*) mit dem *Multifunction Vehicle Bus* (MVB) ermöglicht. Die genaue Struktur der übertragenen Nachrichten hängt dabei vom jeweiligen Zugbeeinflussungssystem (LZB/PZB (Deutschland), ATO (UK), ZUB (Schweiz)) ab. Auch alternative Schnittstellen wie *ProfiNet*, *ProfiBus* oder Ethernet (*Train Real-Time Data Protocol* (TRDP)) können dabei eine Rolle spielen. Für die juristischen On-Board-Daten muss die Kompatibilität mit der FID *Juridical Recording* der European Railway Agency gegeben sein.

Insbesondere werden die folgenden Nachrichtentypen sowie deren Inhalte festgelegt:

- *Themis Message*
- *Request/Response*
- *Transaction*
- *Ordering Message*
- *View Change*
- *Checkpoint*

Nr.	Feld	Typ	Erklärung
1	new_view	64 bit unsigned integer	Neue *View*-Nummer
2	checkpoint	64 bit unsigned integer	Sequenznummer des letzten (durch eine *Checkpoint*-Nachricht) verifizierten *Checkpoints*.
3	checkpoint_proof	*Checkpoint*-Nachrichten	$2f + 1$ verifizierte *Checkpoint*-Nachrichten, die Feld 2 verifizieren.
4	*prepares*	*PrepareProof*	*Beinhaltet m* Mengen von (*Pre*)*Prepare*-Nachrichten (jeweils 1 *Preprepare* und $2f$ dazu passende *Prepare*). *m* ist die Anzahl an *Requests* mit einer Sequenznummer höher als Feld 2.

Tabelle 8.1: Struktur einer Nachricht für einen *View Change*

Eine wichtige neue Funktionalität besteht darin, dass die aktuellen juristischen Daten durch den JBR zur externen Speicherung und Analyse in eine oder mehrere Cloudsysteme hochgeladen werden. Die Cloud an sich agiert dabei als separater, verifizierter JBR-Client. Die Nachrichten zum Austausch sind via *Protocol Buffers* (auch *protobuf* genannt) serialisiert und werden asymmetrisch verschlüsselt ausgetauscht. Dazu werden die zu dieser Kommunikation benötigten Nachrichten definiert:

- *Export*-Nachricht
- *Checkpoint Proof*
- *Checkpoint Blocks*
- *Current Checkpoint*

Ergänzend dazu wurde für die Infrastrukturseite der Use Case sowie die Systemspezifikation erstellt. Da sich die Protokolle der DLTs an ihren Schnittstellen nicht wesentlich unterscheiden, kann auf dieser Grundlage eine FIS für Infrastruktur leicht abgeleitet werden.

8.2 Abstimmung mit Normungsvorhaben

Ein Teilziel des Projektes war das Schaffen einer Grundlage für künftige Normungsvorhaben. Dies ist durch die Systemspezifikation sowie die FIS für den JBR auf Fahrzeugseite sowie die Systemspezifikation für die Infrastrukturseite erfolgt. Die Use Cases und Referenzarchitekturen entsprechen inhaltlich den Vorgaben der o. g. Normen, die FIS für die Fahrzeugseite geht weit darüber hinaus. Damit können diese Ergebnisse direkt in die entsprechenden Normungsprozesse eingebracht werden, z. B. von der DB AG, die in allen maßgeblichen Gremien weitreichenden Einfluss besitzt.

Außerdem erfolgte ein Austausch mit der DIN zur „Normungsroadmap KI“. Durch die Nutzung bereits standardisierter juristischer Daten in einem neuen technologischen Kontext erfüllt das *RailChain*-Projekt dabei insbesondere die Handlungsempfehlung zur Umsetzung von Datenreferenzmodellen für die Interoperabilität von KI-Systemen. Zwar ist die explizite Nutzung von KI nicht Teil des *RailChain*-Projektes, durch die Nutzung des JBR verbessert sich jedoch der Zugang zu Daten, was in der Zukunft eine leichtere Auswertung – auch mit KI-Methoden – möglich macht. Darüber hinaus ermöglicht eine digitale Lösung wie der JBR eine Erweiterung der aufgezeichneten Daten, sowohl zug- als auch streckenseitig. Bei den bisher eingesetzten hardwarebasierten Lösungen ist dies nicht der Fall. Zusammenfassend legt das Projekt den Grundstein für erweiterte Datenanalyse mit Hilfe von neuen Methoden, u. a. aus dem Bereich der künstlichen Intelligenz. Dies kann auf lange Sicht zu einer Wertsteigerung der Daten führen.

9 Zusammenfassung und Ausblick

Das Projekt *RailChain* hat einen verteilten *Juridical Blockchain Recorder* (JBR), der auf einem echtzeitfähigen verteilten Konsensprotokoll basiert, entworfen, implementiert, experimentell evaluiert und als *distributed ledger* an Bord des *advanced TrainLab*s von DB Systemtechnik (ICE-TD 605 017) umgesetzt.

Die erfolgreiche Nachweisführung zeigt mögliche Perspektiven der Technologie auf. Neben der hier untersuchten Funktionalität *Juridical Recorder*, wäre auch ein Einsatz in der Leit- und Sicherungstechnik bzw. Stellwerkstechnik, innerhalb des Eisenbahnbetriebes denkbar. Aber auch neue Geschäftsmodelle könnten sich etablieren, da die Datenerfassung und Verkettung in Echtzeit bei niedrigen Kosten andere Verrechnungsformen ermöglichen.

Besonders hervorzuheben ist die Zusammenarbeit der Projektpartner. Es hat eine Vernetzung der Zusammenarbeit in allen Arbeitspaketen stattgefunden. Das unter dem Aspekt eines Projekts in Zeiten der COVID-Pandemie umso bemerkenswerter ist.

Der Zugang zum Projekt *advanced TrainLab* der Deutschen Bahn und die Kooperation mit dem assoziierten Partner DB Systemtechnik haben die Erreichung dieses Zieles des Projektes überhaupt möglich gemacht und an dieser Stelle sei es erlaubt einen herzlichen Dank an diese beiden Partner des Projektes auszusprechen.

Die Demonstration und Präsentation der Ergebnisse wurden sowohl von Fachpublikum wie auch von den Besuchern, der diversen Veranstaltungen, äußert positiv aufgenommen.

Wir freuen uns über den Abschluss eines sehr schönen Projektes.

Literaturverzeichnis

[1] M. Andersch. „Fehlertoleranz und Zeitverhalten einer Logging-Blockchain unter Verwendung von "Tiered Consistency"". Bachelorarbeit. Professur für Betriebssysteme und Middleware, Hasso-Plattner-Institut, Universität Potsdam, 2020.

[2] J. Braband, R. Kapitza, A. Polze und I. Schwarzer. „RailChain – Anwendung von Distributed-Ledger-Technologien im Bahnbetrieb". In: *Signal + Draht* 114 (Juni 2022), Seiten 16–23. ISSN: 0037-4997.

[3] J. Braband und H. Schäbe. „Zuverlässigkeit von Verteiltem Juridical Recording". In: *Signal + Draht* 113 (Sep. 2021), Seiten 60–67. ISSN: 0037-4997.

[4] Bundesministerium für Digitales und Verkehr (BMDV). *mFUND – Unsere Förderung für die Mobilität der Zukunft*. Besucht am 23.12.2022. Dez. 2022. URL: https://bmdv.bund.de/DE/Themen/Digitales/mFund/Ueberblick/ueberblick.html.

[5] D. Burnett, G. Noble, K. D. Hartog, M. Sporny, B. Zundel und D. Longley. *Verifiable Credentials Data Model v1.1*. W3C Recommendation. World Wide Web Consortium, März 2022.

[6] M. Castro, B. Liskov u. a. „Practical byzantine fault tolerance". In: *OsDI*. Band 99. 1999, Seiten 173–186.

[7] M. Cebe, E. Erdin, K. Akkaya, H. Aksu und S. Uluagac. „Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles". In: *IEEE Communications Magazine* (2018), Seiten 50–57.

[8] CONNECTA project. *Contributing to Shift2Rail's next generation of high capable and safe tcms and brakes – D3.1 – Drive-by-Data Requirement Specification*. 2017.

[9] DB, NS, ÖBB, SNCF und SBB. *OCORA Publications*. Besucht am 23.12.2022. 2020. URL: https://github.com/OCORA-Public/Publications.

[10] *DIN SPEC 3103:2019-06: Blockchain and distributed ledger technologies in application scenarios for Industry 4.0*. Norm. Juni 2019.

[11] *EN 50121-1: Railway applications – Electromagnetic compatibility Part 1: General*. Norm. Nov. 2017.

[12] *EN 50125-1: Railway applications – Environmental conditions for equipment - Part 1: Rolling stock and on-board equipment*. Norm. Nov. 2014.

[13] Entwickler:innen der Ansible-Rolle *tsn*. *tsn - master - Lukas Pirl / ansible-roles-specific*. Besucht am 23.12.2022. URL: https://github.com/optimeas/vmdb2-image-smartmini-tx6.

[14] Entwickler:innen des Projekts *TCNOPen*. *TCNOpen*. Besucht am 23.12.2022. URL: https://sourceforge.net/projects/tcnopen/.

[15] Entwickler:innen des Projekts *vmdb2-image-smartmini-tx6*. *optimeas/vmdb2-image-smartmini-tx6*. Besucht am 23.12.2022. URL: https://github.com/optimeas/vmdb2-image-smartmini-tx6.

[16] Entwickler:innen des Projekts *ZugChain*. *ibr-ds/zugchain*. Besucht am 23.12.2022. URL: https://github.com/ibr-ds/zugchain.

[17] ERTMS Users Group. *ERTMS/ETCS – FIS Juridical Recording*. Technischer Bericht ETCS SUBSET-027. Eisenbahnagentur der Europäischen Union, 2016.

[18] EULYNX-Partner. *Interface definition and specification SMI*. Technischer Bericht Eu.Doc.76. EULYNX-Initiative, 2020.

[19] EULYNX-Partner. *Interface definition SCI*. Technischer Bericht Eu.Doc.92. EULYNX-Initiative, 2020.

[20] EULYNX-Partner. *Interface definition SDI*. Technischer Bericht Eu.Doc.77. EULYNX-Initiative, 2020.

[21] EULYNX-Partner. *Interface specification SCI Generic*. Technischer Bericht Eu.Doc.93. EULYNX-Initiative, 2020.

[22] EULYNX-Partner. *Interface specification SDI Generic*. Technischer Bericht Eu.Doc.94. EULYNX-Initiative, 2020.

[23] EULYNX-Partner. *Maintenance and data management specification*. Technischer Bericht Eu.Doc.18. EULYNX-Initiative, 2020.

[24] EULYNX-Partner. *Specification of Point of Service-Signalling*. Technischer Bericht Eu.Doc.100. EULYNX-Initiative, 2020.

[25] F. Gohla. „Minimalistische Architektur für eine Logging-Blockchain“. Bachelorarbeit. Professur für Betriebssysteme und Middleware, Hasso-Plattner-Institut, Universität Potsdam, 2020.

[26] H. Guo, E. Meamari und C.-C. Shen. „Blockchain-inspired Event Recording System for Autonomous Vehicles“. In: *2018 1st IEEE international conference on hot information-centric networking (HotICN)*. IEEE. 2018, Seiten 218–222.

[27] A. Guy, M. Sabadello, M. Sporny und D. Reed. *Decentralized Identifiers (DIDs) v1.0*. W3C Recommendation. World Wide Web Consortium, Juli 2022.

[28] M. Hartong, R. Goel und D. Wijesekera. „Protection and Recovery of Railroad Event Recorder Data“. In: *IFIP Digital Forensics*. 2008.

[29] Hasso-Plattner-Institut der Universität Potsdam. *HPI-Studierende erforschen den Einsatz von Blockchains in Zügen*. Pressemitteilung. Juli 2022.

[30] R. W. van der Heijden, F. Engelmann, D. Mödinger, F. Schönig und F. Kargl. „Blackchain: Scalability for Resource-Constrained Accountable Vehicle-to-X Communication“. In: *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. 2017, Seiten 1–5.

[31] *IEC 60721-3-3:2019: Classification of environmental conditions - Part 3-3: Classification of groups of environmental parameters and their severities - Stationary use at weatherprotected locations*. Norm. 2019.

[32] *IEC 61375-1:2012: Electronic railway equipment – Train communication network (TCN) – Part 1: General architecture*. Norm. 2012.

[33] *IEC 61375-2-3:2015: Electronic railway equipment – Train communication network (TCN) – Part 2-3: TCN communication profile*. Norm. 2015.

[34] *IEC 61375-3-1:2012: Electronic railway equipment – Train communication network (TCN) – Part 3-1: Multifunction Vehicle Bus (MVB)*. Norm. 2012.

[35] *IEC 62625-1:2014: Electronic railway equipment – On board driving data recording system - Part 1: System specification*. Norm. 2014.

[36] „IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks“. In: *IEEE Standard 802.1Q-2018* (2018).

[37] „IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 25: Enhancements for Scheduled Traffic“. In: *IEEE Standard 802.1Qbv-2015* (2015).

[38] „IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications“. In: *IEEE Standard 802.1AS-2020* (2020).

[39] „IEEE Standard for Local and metropolitan area networks – Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams“. In: *IEEE Standard 802.1Qav-2009* (2009).

[40] *Blockchain and distributed ledger technologies – Reference architecture*. Norm. International Organization for Standardization (ISO), Feb. 2022.

[41] *Blockchain and distributed ledger technologies – Use cases*. Norm. International Organization for Standardization (ISO), Okt. 2022.

[42] M. Kuperberg. „Scaling a Blockchain-based Railway Control System Prototype for Mainline Railways: a Progress Report“. In: *ArXiv* abs/2103.08304 (2021).

[43] M. Kuperberg, D. Kindler und S. Jeschke. „Are Smart Contracts and Blockchains Suitable for Decentralized Railway Control?“ In: *ArXiv* (2019).

[44] G. Muniandi. „Blockchain-enabled Virtual Coupling of Automatic Train Operation Fitted Mainline Trains for Railway Traffic Conflict Control“. In: *IET Intelligent Transport Systems* 14.6 (2020), Seiten 611–619.

[45] F. Naser. „The Potential Use of Blockchain Technology in Railway Applications: An Introduction of a Mobility and Speech Recognition Prototype“. In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE. 2018, Seiten 4516–4524.

[46] C. Oham, S. S. Kanhere, R. Jurdak und S. Jha. „A Blockchain Based Liability Attribution Framework for Autonomous Vehicles“. In: *ArXiv* (2018).

[47] J. Preece und J. Easton. „A Review of Prospective Applications of Blockchain Technology in the Railway Industry“. In: *Preprint submitted to International Journal of Railway Technology* (2018).

[48] J. D. Preece und J. M. Easton. „Blockchain Technology as a Mechanism for Digital Railway Ticketing“. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, Seiten 3599–3606.

[49] S. Rüsch, K. Bleeke und R. Kapitza. „Themis: An Efficient and Memory-Safe BFT Framework in Rust: Research Statement“. In: *Proceedings of the 3rd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. 2019, Seiten 9–10.

[50] S. Rüsch, K. Bleeke, I. Messadi, S. Schmidt, A. Krampf, K. Olze, S. Stahnke, R. Schmid, L. Pirl, R. Kittel, A. Polze, M. Franz, M. Müller, L. Jehl und R. Kapitza. „ZugChain: Blockchain-Based Juridical Data Recording in Railway Systems“. In: *52nd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2022, Seiten 67–78.

[51] C.-S. Shih, W.-Y. Hsieh und C.-L. Kao. „Traceability for Vehicular Network Real-Time Messaging Based on Blockchain Technology.“ In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 10.4 (2019), Seiten 1–21.

[52] G. Wang, Z. J. Shi, M. Nixon und S. Han. „ChainSplitter: Towards Blockchain-Based Industrial IoT Architecture for Supporting Hierarchical Storage“. In: *2019 IEEE International Conference on Blockchain (Blockchain)* (2019), Seiten 166–175.

[53] R. White, G. Caiazza, A. Cortesi, Y. Im Cho und H. I. Christensen. „Black Block Recorder: Immutable Black Box Logging for Robots via Blockchain“. In: *IEEE Robotics and Automation Letters* 4.4 (2019), Seiten 3812–3819.

[54] X. Xu, I. Weber und M. Staples. *Architecture for Blockchain Applications*. Springer, 2019.

[55] L. Zhang, Y. Huang und T. Jiang. „High-speed Railway Environmental Monitoring Data Identity Authentication Scheme Based on Consortium Blockchain“. In: *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*. 2019, Seiten 7–13.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren/Redaktion
151	978-3-86956-547-7	**HPI Future SOC Lab – Proceedings 2018**	Christoph Meinel, Andreas Polze, Karsten Beins, Rolf Strotmann, Ulrich Seibold, Kurt Rödszus, Jürgen Müller
150	978-3-86956-546-0	**openHPI : 10 Jahre MOOCs am Hasso-Plattner-Institut**	Christoph Meinel, Christian Willems, Thomas Staubitz, Dominic Sauer, Christiane Hagedorn
149	978-3-86956-545-3	**Implementing a crowd-sourced picture archive for Bad Harzburg**	Rieke Freund, Jan Philip Rätsch, Franziska Hradilak, Benedikt Vidic, Oliver Heß, Nils Lißner, Hendrik Wölert, Jens Lincke, Tom Beckmann, Robert Hirschfeld
148	978-3-86956-544-6	**openHPI : 10 Years of MOOCs at the Hasso Plattner Institute**	Christoph Meinel, Christian Willems, Thomas Staubitz, Dominic Sauer, Christiane Hagedorn
147	978-3-86956-533-0	**Modeling and formal analysis of meta-ecosystems with dynamic structure using graph transformation**	Boris Flotterer, Maria Maximova, Sven Schneider, Johannes Dyck, Christian Zöllner, Holger Giese, Christelle Hély, Cédric Gaucherel
146	978-3-86956-532-3	**Probabilistic metric temporal graph logic**	Sven Schneider, Maria Maximova, Holger Giese
145	978-3-86956-528-6	**Learning from failure : a history-based, lightweight test prioritization technique connecting software changes to test failures**	Falco Dürsch, Patrick Rein, Toni Mattis, Robert Hirschfeld
144	978-3-86956-526-2	**Die HPI Schul-Cloud – Von der Vision zur digitalen Infrastruktur für deutsche Schulen**	Christoph Meinel, Catrina John, Tobias Wollowski, HPI Schul-Cloud Team
143	978-3-86956-531-6	**Invariant analysis for multi-agent graph transformation systems using k-induction**	Sven Schneider, Maria Maximova, Holger Giese

ISBN 978-3-86956-550-7
ISSN 1613-5652