Entwicklung eines industriellen Blockchain-Netzwerkes

Erik Neumann
Hochschule Mittweida, Technikumplatz 17, 09648 Mittweida

Mit der zunehmenden Vernetzung von Unternehmen wächst auch das Potenzial für Cyberangriffe, Spionage und Sabotage in Produktionsnetzwerken. Netzwerke, die auf Blockchain-Technologien aufbauen, können einige dieser Risiken abmildern, insbesondere solche, die Datenmanipulation betreffen. Dieses Paper befasst sich mit der Architektur und Implementierung eines unternehmensübergreifenden Blockchain-Netzwerks zur manipulationssicheren und ausfallsicheren Speicherung von produktionsbezogenen Daten und deren Verteilung innerhalb eines globalen Netzwerks. Dazu werden zunächst die Anforderungen an ein solches System erläutert. Darauf aufbauend wird die Architektur eines Blockchain-Knotens beschrieben und der Nutzen des Systems anhand eines Anwendungsfalls dargestellt.

As companies become more interconnected, the potential for cyberattacks, espionage, and sabotage in production networks continues to grow. Networks building on blockchain technologies can mitigate some of these risks, especially those concerning data manipulation. This paper details the architecture and implementation of a cross-company blockchain network for the tamper-proof and resilient storage of production-related data and its distribution within a global network. To this end, the requirements for such a system are first explained. Based on this, the architecture of a blockchain node is described and the utility of the system is presented via a use case.

1. Einleitung

Der technologische Fortschritt der letzten Jahre hat es Unternehmen ermöglicht, ihre Produktionssysteme schrittweise zu digitalisieren und zu vernetzen. Mithilfe dieser Vernetzung können mehrere Unternehmen entlang einer Wertschöpfungskette ihre Produktion aufeinander anpassen und effizienter zusammenarbeiten [1]. Durch die stärkere Vernetzung und Digitalisierung vergrößert sich jedoch auch die Gefahr durch Cyberkriminalität. So wurde zwischen 2019 und 2020 ein rund 20-prozentiger Anstieg bei den Delikten der „*Datenveränderung*“ und „*Fälschung beweiserheblicher Daten*“ [1] verzeichnet, dabei waren die Ziele zumeist große Unternehmen [2][3].

Insbesondere die Manipulation von produktionsnahen Daten stellt für die Sicherheit von kritischen Infrastrukturen eine erhöhte Gefahr dar. Werden beispielsweise Produktionsdaten manipuliert, kann die Nachverfolgbarkeit von Fehlern innerhalb der Produktion zu einem späteren Zeitpunkt nicht mehr gewährleistet werden. Dies kann zu einer Gefährdung der Bevölkerung führen, falls Fehler beispielsweise bei Automobilteilen oder pharmazeutischen Produkten auftreten.

Um eine Lösung für dieses Problem zu erforschen, wurde das Projekt „safe-UR-chain“ 2019 vom Bundesministerium für Bildung und Forschung gefördert [4], darin arbeiten Unternehmen mit Expertise in der Fertigungsindustrie mit Forschungseinrichtungen[2] gemeinsam an einem Blockchain-Simulator zur Erprobung innerhalb eines Wertschöpfungsnetzwerkes. Die Entwicklung des Blockchain-Netzwerkes wird dabei von der Hochschule Mittweida getragen. Dieses Netzwerk soll es ermöglichen, Daten manipulationssicher speichern zu können. Dafür sollen einzelne Unternehmen jeweils private Blockchains verwenden, die sich gegenseitig absichern. Dabei werden die unternehmensinternen Blockchains regelmäßig die Hashes ihrer aktuellen Blöcke austauschen. Mithilfe dieser Hashes kann die Existenz von Daten innerhalb der jeweiligen Blockchains später belegt werden, ohne dass die gesamten Daten preisgegeben werden müssen.

2. Anforderungen

Da die Blockchain-Software innerhalb eines industriellen Umfeldes zum Einsatz kommen soll, muss sie bestimmten Anforderungen gerecht werden. Diese Anforderungen wurden insbesondere in Zusammenarbeit mit den Industriepartnern ausgearbeitet. Einige der Anforderungen wurden auf Basis der unternehmensinternen Infrastruktur entwickelt und betreffen Limits, die beispielsweise durch die Konfiguration von Firewalls und dem Firmennetzwerk entstehen. Andere betreffen die Kommunikation innerhalb des Blockchain-Netzwerkes, sowie die Blockchain selbst. Diese Anforderungen entspringen hauptsächlich Überlegungen über die Sicherheit des Systems, sowie der Grundidee, einzelne Netzwerkteilnehmer nur mit für sie unbedingt notwendigen Daten in Kontakt kommen zu lassen. Die wesentlichen Anforderungen an das System sind im Nachfolgenden aufgeführt:

[1] Deliktsbezeichnungen gekürzt

[2] https://safe-ur-chain.de/about

- Die Netzwerkinterne Kommunikation erfolgt über TCP/IP
- Die Kommunikation über Unternehmensgrenzen hinaus erfolgt über HTTPS
- Das Blockchain-Netzwerk baut sich selbstständig auf
- Beim Netzwerkaufbau kommen keine Broadcast-nachrichten zum Einsatz
- Nachrichten innerhalb des Netzwerkes werden signiert
- Neue Nodes synchronisieren die Blockchain automatisch
- Das Fehlen von Nutzdaten darf den Block-Hash nicht beeinflussen

3. Node-Architektur

Auf Basis dieser Anforderungen wurde die Software der Blockchain-Nodes entwickelt. Diese Software teilt sich in verschiedene Module, die den Fokus des restlichen Kapitels bilden.

3.1. Blockchain

Das zentrale Modul der Node-Software bildet Funktionen der Blockchain ab. Es wurde auf den drei Ebenen der Transaktionen, Blöcke, sowie der eigentlichen Blockchain implementiert.

Transaktionen stehen in der untersten Ebene der Blockchain-Hierarchie, sie enthalten die eigentlichen Nutzdaten, die vom Netzwerk gespeichert werden sollen. Um jegliche Art von Daten aufnehmen zu können, bieten Transaktionen das *payload*-Feld, in dem Daten als Bytes abgelegt werden können. Der Datentyp wird vom *contents*-Feld repräsentiert und für die Deserialisierung der Daten verwendet. Um die Herkunft der Daten anzuzeigen, wird ihr Hash-Wert von der Node, die sie aufgenommen hat signiert und im *signed_hash*-Feld abgespeichert. Mit dieser Information können die Nutzdaten nun auch aus der Transaktion entfernt, später aber auch wieder zugeordnet werden. Transaktionen, die keine Nutzdaten mehr enthalten werden als *stripped* bezeichnet und können zur Einsparung von Bandbreite bzw. für die Geheimhaltung sensibler Daten verwendet werden. Zusätzlich zum signierten Hash erhält jede Transaktion zum späteren Auffinden eine eindeutige Identifikationsnummer. Diese wird deterministisch aus dem signierten Hash der Nutzdaten, sowie dem Zeitstempel der Transaktion erzeugt (dieser ist in den Metadaten enthalten). Zusätzlich können im Feld *tags* Schlagwörter hinterlegt werden, die eine effiziente Suche nach Transaktionen ermöglichen. Sie geben den Nutzern des Systems eine Möglichkeit, Transaktionen beispielsweise mit intern verwendeten Teilenummern oder Maschinenkennungen zu versehen.

```rust
pub struct Transaction {
    pub id: Vec<u8>,
    pub tags: Vec<String>,
    pub signed_hash: SignedData,
    pub contents: TransactionV1Contents,
    pub payload: Vec<u8>,
    pub stripped: bool,
    pub meta: TransactionMetaData,
}
```

Abbildung 1: Datenstruktur für eine Transaktion

Mehrere Transaktionen werden zu Blöcken zusammengefasst, dafür wird eine Merkle-Tree-Struktur [5] verwendet, bei der in unterster Ebene die IDs der Transaktionen stehen, da diese IDs unabhängig davon sind, ob die Transaktion Nutzdaten enthalten oder nicht, erhält der Merkle-Tree immer den gleichen Root-Hash, selbst wenn einige oder alle der Transaktionen frei von Nutzdaten sind.

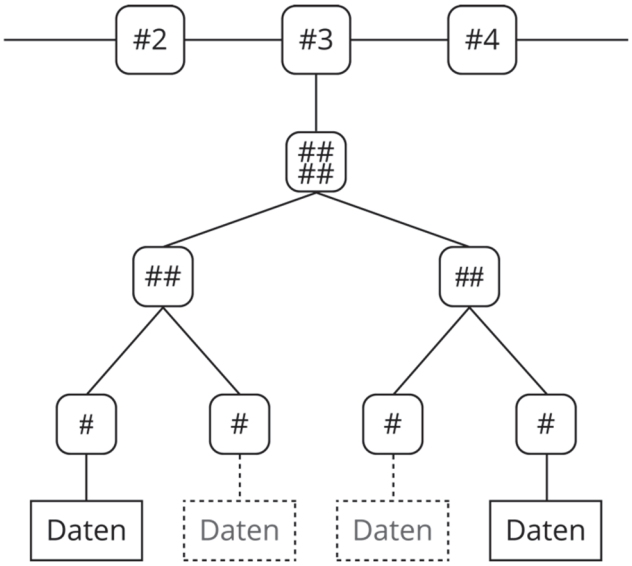

Abbildung 2: Merkle-Tree mit einigen fehlenden Nutzdaten und der Verknüpfung von dessen Root-Hash mit einem Block

Zusätzlich zu dieser Baumstruktur enthalten Blöcke auch *Header*, diese Datenstruktur enthält einige Metadaten, wie den Zeitstempel und die Block-Höhe, sowie Felder, die eingesetzt werden, um die Blockerzeugung nach den Regeln verschiedener Konsensverfahren zu erlauben. So können die Felder *difficulty*, *nonce* und *signatures* in unterschiedlichen Konsensverfahren verschiedene Rollen einnehmen. Dies ermöglicht es den Unternehmen, in ihrer lokalen Blockchain ein Konsensverfahren einzusetzen, das speziell auf ihren Anwendungsfall angepasst ist. Zum Testen dieser Funktionalität wurden *Proof of Work*, *Proof of Elapsed Time*, sowie ein Verfahren, bei dem Blöcke valide werden, wenn sie von ausreichend vielen Nodes signiert wurden, implementiert.

```rust
pub struct BlockHeader {
    pub timestamp: u128,
    pub previous_digest: Vec<u8>,
    pub difficulty: Difficulty,
    pub nonce: Vec<u128>,
    pub height: usize,
    pub merkle_root: Vec<u8>,
    pub signatures: Vec<SignedData>,
}

pub struct Block {
    pub header: BlockHeader,
    pub data: MerkleTree,
    hash: Option<Vec<u8>>,
}
```

Abbildung 3: Datenstrukturen für einen *Block* und dessen *Header*

Die Datenstruktur für Blöcke beinhaltet zusätzlich ein Feld für den Block-Hash. Dieser wird im Netzwerk nicht übertragen und von jeder Node selbst erzeugt, indem der Hash des Headers gebildet wird (in diesem ist der Root-Hash des Merkle-Trees enthalten).

Die oberste Ebene der Blockchain-Struktur wird als Baum-Struktur abgebildet, in der neue Blöcke an ihren jeweiligen Vorgänger angehangen werden. Um die korrekte Blockchain zu erzeugen, bzw. um Forks aufzulösen, verwendet diese Baumstruktur je nach Konsensverfahren eine Scoring-Funktion, mit der jeder Block eine Punktzahl erhält, diese Punktzahl, sowie die Summe der Punktzahlen der Vorgängerblöcke wird von jedem Element der Baumstruktur gespeichert. Um nun die korrekte/längste/gültige Kette zu bilden, wird das Ende mit der höchsten Punktzahl gesucht und dessen Vorgänger werden aufgelöst. Die Baumstruktur kann ebenfalls verwaiste Blöcke[3] aufnehmen, diese werden automatisch in die Kette integriert, sobald ihr Vorgänger hinzugefügt wird. Diese Funktion ist rekursiv implementiert, damit auch Ketten von verwaisten Blöcken korrekt an die Blockchain angehangen werden.

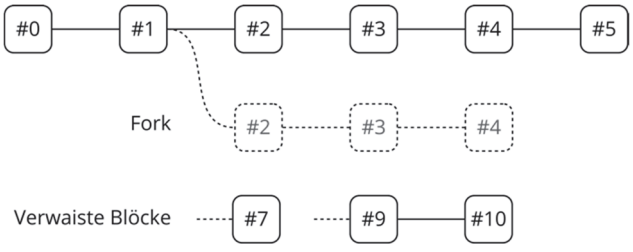

Abbildung 4: Baumstruktur zur Speicherung von Blöcken mit der „korrekten“ Kette (oben), einer Fork (mittig) und verwaisten Blöcken (unten)

Um diese Datenstrukturen zu speichern und für die Software verfügbar zu machen, wurde ein Speicher auf Basis einer Dateisystem-Datenbank entwickelt, der die Ebenen der Blockchain-Struktur widerspiegelt und einen Datenabruf in konstanter Zeit ermöglicht. Zusätzlich wurde aus dieser Speicherstruktur ein Interface abgeleitet, das es Unternehmen ermöglicht, ihre eigenen Speicherlösungen an die Node-Software anzuschließen.

3.2. Datenaufnahme

Das Datenaufnahmemodul bietet ein generisches Interface mit dem Daten in das System eingespeist werden können. Über dieses Interface können Daten entweder als reine Rohdaten oder als Bündel aus Roh-und Metadaten übertragen werden. Die eingegangenen Datensätze werden dann von der Blockchain-Node signiert und in Transaktionen eingearbeitet, die in die Blockchain geschrieben werden. Unternehmen können mithilfe dieses Interfaces ihre eigenen Protokolle mit beliebiger Logik umsetzen.

Für dieses Interface wurden bereits drei Implementierungen geschrieben. Die erste erlaubt die Aufnahme von Daten direkt aus dem Dateisystem. Dabei wird lediglich ein Verzeichnis auf neue Dateien überwacht, sobald diese erkannt werden, wird ihr Inhalt eingelesen und daraus Transaktionen erstellt. Die zweite Implementierung wurde als Gegenstelle zu einem proprietären Kommunikationsprotokoll von einem der Projektpartner entwickelt. Dieses Protokoll erlaub die Aufnahme von verschlüsselten Nutzdaten mit Metadaten direkt von den Maschinen des Partners. Mit dem Interface für die Datenaufnahme lassen sich beliebige Netzwerkprotokolle einbauen, so ermöglicht die dritte Implementierung die Übertragung von Dateien über HTTP, damit können Nutzdaten beispielsweise über den Dateiupload in einem Webbrowser an die Blockchain-Node übergeben werden.

3.3. Netzwerk

Die bisher beschriebenen Module bieten die grundlegenden Datenstrukturen für die Blockchain, sowie eine Möglichkeit, Daten in diese aufzunehmen. Das Netzwerk-Modul ermöglicht es, diese Funktionalitäten im Netzwerk zu verteilen. So werden aufgenommene Daten in das Netzwerk geschickt und dort von Nodes die Blöcke erzeugen aggregiert. Die erzeugten Blöcke müssen ihrerseits auch im Netzwerk verteilt und auf den empfangenden Nodes an die Blockchain angefügt werden.

Um diese Funktionen bereitzustellen, bietet das Netzwerkmodul eine interne API, über die andere Prozesse ausgehende Nachrichten an das Netzwerkmodul übergeben und eingehende Nachrichten vom Netzwerkmodul abrufen können. Die Logik, die innerhalb des Moduls für den Aufbau und Erhalt des Netzwerkes zuständig ist, wurde vom restlichen System abgekapselt.

[3] Blöcke die vor ihrem Vorgänger empfangen wurden

Der Netzwerkaufbau darf laut den bereits genannten Anforderungen nicht auf Broadcast-Nachrichten aufbauen. Darum erhalten Nodes beim ersten Start eine Liste von anderen Nodes, die mit hoher Wahrscheinlichkeit online sind. Diese Nodes werden als Seed-Nodes bezeichnet. Das Netzwerkmodul einer neu gestarteten Node beginnt damit, Nachrichten an die Seed-Nodes zu schicken, die eine Anfrage nach deren Liste von bekannten Nodes enthält. Nach Erhalt dieser Liste, schickt das Netzwerkmodul die gleiche Anfrage auch an die nun bekannten Nodes, bis eine gewisse Mindestanzahl an bekannten Nodes erreicht ist.

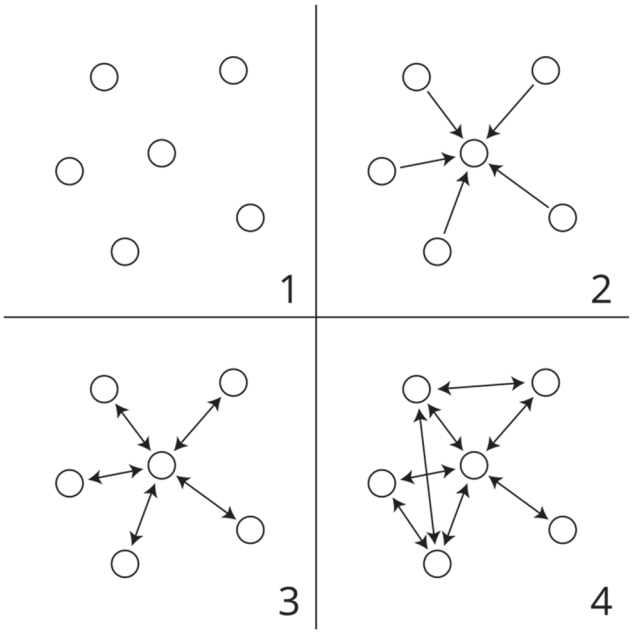

Abbildung 5: Schrittweiser Netzwerkaufbau mit Hilfe einer Seed Node; 1) Keine Verbindungen; 2) Unidirektionale Verbindungen zur Seed-Node (Anfragen); 3) Erste bidirektionale Verbindungen zwischen Nodes; 4) Verbindungen zwischen Nodes die sich durch Informationen der Seed Node finden konnten

Nodes streben danach, eine dauerhafte Verbindung zu einigen anderen Nodes aufrechtzuerhalten. Diese werden zufällig aus der Liste der bekannten Nodes ausgewählt und regelmäßig erneuert. Damit soll die zufällige oder böswillige Abschottung von Nodes verhindert werden.

3.4. Kryptographie

Alle Transaktionen und Netzwerk-Nachrichten werden von den Nodes signiert. Dafür erhalten Nodes je ein Schlüsselpaar aus einem öffentlichen und privaten Schlüssel. Diese Schlüssel werden in der aktuellen Implementierung durch ein hierarchisch-deterministisches Verfahren [6] erzeugt und beispielsweise durch einen Administrator auf die Nodes verteilt. Der erweiterte öffentliche Schlüssel ist allen Nodes bekannt, damit können neue Nodes dem Netzwerk beitreten, ohne dass ihr öffentlicher Schlüssel bekanntgegeben werden muss, andere Nodes können dieses mithilfe des erweiterten öffentlichen Schlüssels, sowie der Kennung der neuen Node erzeugen.

Die Logik für die Kryptographie wurde in ein separates Modul gekapselt, dieses bietet Funktionen zum Signieren, Hashen und Verschlüsseln von Daten. Die Verschlüsselung wurde dabei über ein das *Elliptic Curve Integrated Encryption Scheme* [7] umgesetzt. In diesem hybriden Verschlüsselungsverfahren wird der öffentliche Schlüssel des Empfängers zum Verschlüsseln eines temporären, symmetrischen Schlüssels genutzt, mit dem die Nutzdaten verschlüsselt werden. Damit können Nodes ohne zusätzliches Setup verschlüsselte Nachrichten untereinander austauschen.

3.5. Processing

Innerhalb der Node fallen viele verschiedene Aufgaben an. So müssen beispielsweise empfangene Blöcke verarbeitet, fehlende Blöcke angefragt und Transaktionen erstellt werden. Diese Aufgaben werden innerhalb der Node vom Processing-Modul verarbeitet. Dieses Modul gibt Warteschlangen für Aufgaben frei, die von einer konfigurierbaren Anzahl von *Worker*-Threads abgearbeitet und die anderen Module weitergegeben werden.

Zusätzlich hat dieses Modul die Aufgabe, regelmäßige Überprüfungen zum Zweck von Instandhaltungsarbeiten der lokalen Blockchain durchzuführen. Beispielsweise werden lokal erzeugte Transaktionen so lange gespeichert, bis sie in der Blockchain des Netzwerkes auffindbar sind. Falls Transaktionen nach einer gewissen Zeit nicht in die Blockchain aufgenommen wurden, reiht das Modul eine erneute Verteilung im Netzwerk ein. Zusätzlich wird regelmäßig geprüft, ob die lokale Blockchain auf dem aktuellen Stand des Netzwerkes ist, dafür werden Blockchain-Status Nachrichten im Netzwerk, sowie das Alter des aktuellsten lokalen Blockes herangezogen. Sollte festgestellt werden, dass die lokale Blockchain nicht mehr aktuell ist, werden Anfragen an das Netzwerk eingeleitet. Über diesen Mechanismus synchronisieren sich auch Nodes, die zum ersten Mal gestartet werden.

3.6. Architektur

Die in diesem Kapitel beschriebenen Module arbeiten in sechs Funktionsgruppen zusammen, die gemeinsam die Blockchian-Node bilden:

Gruppe	**Aufgabe(n)**
Netzwerk	Netzwerkverwaltung; Kommunikation mit anderen Nodes
Blockchain	Zusammenfassung von Blöcken zur Blockchain
Blockerzeuger	Sammlung von Transaktionen; Erzeugung neuer Blöcke
Speicher	Speicherung und Bereitstellung von Blockchain, Blöcken und Transaktionen
Datenaufnahme	Entgegennahme von Roh- und Metadaten

Processing	Bearbeitung von Aufgaben und Umsetzung von Datenströmen zwischen den anderen Funktionsgruppen

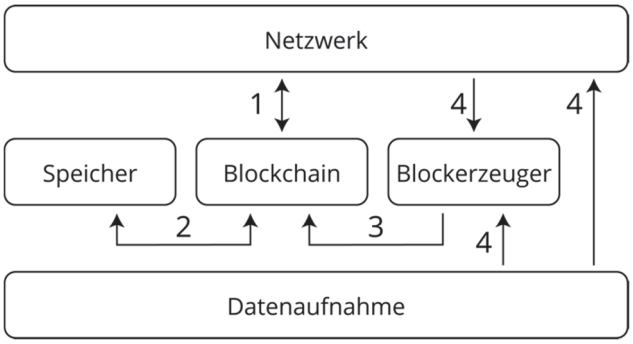

Abbildung 6: Funktionsgruppen und Datenströme zwischen ihnen (Processing); 1) Blöcke und Anfragen nach Daten; 2) Blockchain + alle assoziierten Daten; 3) Blöcke; 4) Transaktionen

4. Netzwerk-Architektur

Das Gesamtnetzwerk lässt sich in zwei Ebenen aufteilen: die lokale Ebene beinhaltet die einzelnen Netzwerke der Unternehmen, die von den Blockchain-Nodes gebildet werden. Und die globale Ebene, über die diese lokalen Netzwerke Informationen austauschen können.

4.1. Lokales Netzwerk

Das lokale Netzwerk besteht aus mehreren Blockchain-Nodes, wie sie in Kapitel 3 beschrieben worden. Um dieses Netzwerk besser an die spezifischen Anforderungen einzelner Unternehmen anpassen zu können, werden bei einigen Nodes die Funktionsgruppe „Blockerzeuger" und „Speicher" abgeschaltet bzw. anders konfiguriert. Durch die Abschaltung des Blockerzeugers können Nodes auf Hardware mit weniger Rechenleistung betrieben werden, da die rechenaufwändige Aufgabe des Zusammensetzens von Transaktionen in Blöcke wegfällt. Und durch die angepasste Konfiguration der Speicher-Funktionsgruppe kann die Speicherung einiger oder aller Transaktionsnutzdaten umgangen werden. Damit können Nodes auf Hardware mit deutlich weniger Speicher betrieben werden.

Es ergeben sich vier verschiedene Node-Typen:

	Blockerzeuger	Speicher
Full Node	Aktiviert	Alle Nutzdaten
Blockerzeuger Node	Aktiviert	Teilweise/Keine Nutzdaten
Archiv Node	Deaktiviert	Alle Nutzdaten
Thin Node	Deaktiviert	Teilweise/Keine Nutzdaten

Durch den gezielten Einsatz dieser Node-Typen können Netzwerke genau an die verfügbare Hardware und den Anwendungsfall angepasst werden. So könnten Thin Nodes innerhalb einer Fabrik direkt an Maschinen die Produktionsdaten aufnehmen und diese an Blockerzeuger Nodes schicken. Die entstehende Blockchain kann dann für den Langzeitspeicher in einem Cluster von Archiv-Nodes gespeichert und von diesen für andere Applikationen zur Verfügung gestellt werden.

4.2. Globales Netzwerk

Auf der Ebene des globalen Netzwerkes tauschen lokale Netzwerke Daten untereinander aus. Dafür werden eine oder mehrere Instanzen eines Message Brokers genutzt, der projektintern aufgrund seiner Funktionsweise als „Post Office" bezeichnet wird. Dieser Broker nimmt verschlüsselte Nachrichten von allen Netzwerken entgegen, sammelt diese und schickt sie auf Anfrage an das Zielnetzwerk der Nachrichten. Die Verschlüsselung der Nachrichten erfolgt über das bereits beschriebene Hybridverfahren, sodass nur Nodes im Zielnetzwerk die Nachrichten entschlüsseln können. Damit wird sichergestellt, dass Nachrichten, die über den Broker versandt werden, nicht von außen mitgelesen werden können.

Die wesentlichen Nachrichten, die Netzwerke austauschen, sind Block-Hashes. Diese werden verwendet, um die Daten innerhalb der lokalen Blockchains von den anderen Teilnehmern des globalen Netzwerkes „gegenzeichnen" zu lassen. Dafür werden empfangene Block-Hashes aus fremden Netzwerken innerhalb eines lokalen Netzwerkes als Transaktion in die Blockchain aufgenommen. Sobald der Hash eines Blockes, der einen anderen Block-Hash als Transaktion enthält in einem anderen Netzwerk in die Blockchain aufgenommen wird, kann an beiden Blockchains bis zu diesem Zeitpunkt keine Veränderung mehr vorgenommen werden, selbst wenn dies vom Konsensverfahren erlaubt würde. Wenn dieses Verfahren von mehreren Unternehmen regelmäßig angewandt wird, werden die Blockchains ineinander „verstrickt". Abbildung 7 veranschaulicht den Prozess.

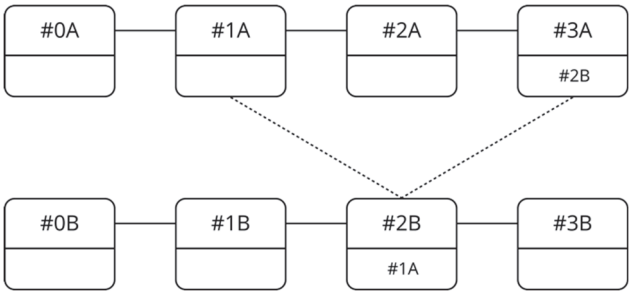

Abbildung 7: „Verstrickte" Blockchains zweier Netzwerke; Die oberen Bereiche der Blöcke enthalten die Blocknummer, die unteren die Hashes von Blöcken externer Blockchains

Der Umgang mit Forks in einem oder mehreren der Netzwerke, sowie ein effizienter Beweis über die Existenz von Daten in einer so verstrickten Blockchain sind Gegenstand laufender Forschung.

5. Anwendungsfall

Ein Anwendungsfall für das System besteht in der Rückverfolgung eines fehlerhaften Produktes. In diesem Anwendungsfall produziert Unternehmen **A** Bauteile, die von Unternehmen **B** verbaut werden. Dabei protokolliert **A** die eigene Produktion sowie eine Qualitätsprüfung auf seiner lokalen Blockchain und gibt die betreffenden Transaktions-IDs mit dem Versand weiter. Unternehmen **B** protokolliert diese IDs beim Wareneingang, und die Montage der eigenen Produkte auf seiner Blockchain.

Sollte im Produkt ein Fehler auftreten, der sich nicht auf die Montage von **B** zurückführen lässt, kann die Preisgabe der Ergebnisse der Qualitätskontrolle von Unternehmen **A** gefordert werden. Die Informationen in dieser Transaktion sind auf Seite von Unternehmen **B** durch die beim Wareneingang erhaltenen Transaktions-IDs nachvollziehbar und können durch die „Verstrickung" der Blockchains zeitlich eingeordnet werden. Wenn festgestellt wird, dass das Bauteil zum Zeitpunkt der Qualitätskontrolle fehlerfrei war, kann davon ausgegangen werden, dass es während des Transportes beschädigt wurde. Andernfalls kann Unternehmen **A** den Fehler weiter zurückverfolgen und für eine Entschädigung sorgen.

6. Ergebnisse

Das beschriebene Netzwerk ist mit Ausnahme der in 4.2 benannten Funktionen, die noch erforscht werden, vollständig implementiert und erfüllt die genannten Anforderungen. Als besonderes Ergebnis ist die Sicherheit zu nennen, mit der die Integrität von Daten innerhalb der Blockchain über Netzwerkgrenzen hinweg belegt werden kann.

Die Node-Software wurde in der Programmiersprache Rust[4] implementiert und umfasst zum Zeitpunkt des Schreibens dieses Papers annähernd 8000 Zeilen Code, die zu einer ausführbaren Binärdatei mit einer Größe von 13,9MB[5] kompiliert werden. Durch die geringe Größe der Binärdatei und den überschaubaren Quellcode kann die Software für viele Anwendungsgebiete verwendet und angepasst werden. Aktuelle Integrationstests zeigen einen Transaktionsdurchsatz von über 100 Transaktionen pro Sekunde.

Im weiteren Projektverlauf soll ein Sicherheitsaudit des Systems, sowie eine Evaluation durch die testweise Anwendung bei den Industriepartnern durchgeführt werden. Erste Tests, darunter die Erprobung eines Testnetzes über mehrere Monate wurden erfolgreich ausgeführt.

Förderhinweis

Das Vorhaben wird mit Mitteln des Bundesministeriums für Bildung und Forschung im Rahmen der Bekanntmachung „Zivile Sicherheit – Kritische Strukturen und Prozesse in Produktion und Logistik" unter den Förderkennzeichen 13N15150 bis 13N15153 gefördert.

Literaturverzeichnis

[1] Siemens. The Digitalization Productivity Bonus, 2017, 7-9.

[2] Bundeskriminalamt. Cybercrime Bundestagsbild 2020, 2021, 3-11.

[3] Statista. Polizeilich erfasste Fälle von Cyberkriminalität im engeren Sinne in Deutschland von 2007 bis 2020 [Online; aufgerufen am 30.08.2021] https://de.statista.com/statistik/daten/studie/295265/umfrage/polizeilich-erfasste-faelle-von-cyberkriminalitaet-im-engeren-sinne-in-deutschland/

[4] Bundesministerium für Bildung und Forschung. Sicherheit und Nachverfolgbarkeit in zivilen Produktions- und Wertschöpfungsnetzwerken durch Blockchain (safe-UR-chain), 2019 [Online; aufgerufen am 30.08.2021] https://www.sifo.de/files/Projektumriss_safe-UR-chain.pdf

[5] Merkle, R., Protocols for Public Key Cryptosystems, 1980 IEEE Symposium on Security and Privacy, 1980, 125–127

[6] Bitcoin Core Team. BIP 32 [Online; aufgerufen am 30.08.2021] https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[7] ECIES Developers. eciesrs [Online; aufgerufen am 30.08.2021] https://github.com/ecies/rs

[4] https://www.rust-lang.org

[5] Die Größe bezieht sich auf die Version für Unix-basierte Betriebssysteme