

Bachelorarbeit (IT)

Teststand für ein vollständig verteiltes und Blockchain-basiertes Handelsnetzwerk für digitale Werte

Autoren	Levi Cailleret Sascha Kyburz
Hauptbetreuung	Dr. Stephan Neuhaus
Datum	11.06.2021

Formular

Eigenständigkeitserklärung

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbstständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Vorname, Name: Levi Cailleret

Ort, Datum: Bütschwil, 09.06.2021

Unterschrift:



Vorname, Name: Sascha Kyburz

Ort, Datum: Kreuzlingen, 09.06.2021

Unterschrift:



Zusammenfassung

Wir untersuchen das Projekt DIVA.EXCHANGE und versuchen dabei mögliche Sicherheitsrisiken auszumachen. DIVA.EXCHANGE ist ein Open Source Projekt mit dem Ziel, mittels einer Blockchain ein vollständig verteiltes Handelsnetzwerk aufzubauen. Die eingesetzte Blockchain basiert auf Hyperledger Iroha und ist primäres Ziel der Forschungsarbeit und unserer Tests.

Auf einem lokalen Netzwerk wird eine Test-Blockchain von DIVA.EXCHANGE aufgebaut. Innerhalb dieser Testumgebung werden mehrere Tests durchgeführt und das Verhalten der Blockchain analysiert. Dabei liegt die Manipulation der Blockchain durch unehrliche Knoten im Fokus. Auch andere mögliche Angriffe auf DIVA.EXCHANGE werden dabei in Betracht gezogen.

Die wichtigsten Resultate zeigen, dass die auf Hyperledger Iroha basierende Blockchain die byzantinische Fehlertoleranz von mehr als zwei Dritteln ehrlichen Knoten nicht einhält. Weiterhin zeigen die Resultate, dass der Konsensalgorithmus YAC, den Hyperledger Iroha verwendet, zu langen Wartezeiten bei der Erstellung von Blöcken innerhalb des Netzwerks führt, wenn nicht alle Netzwerkteilnehmer durchgehend verfügbar sind.

Wir folgern aus den Daten, dass Hyperledger Iroha deutliche Sicherheitsrisiken aufweist, da byzantinische Fehler nicht ausgeschlossen werden können. Dieser Umstand muss verbessert werden, damit Transaktionen von Wertmitteln im öffentlichen Rahmen risikolos stattfinden können. Wir kommen zum Schluss, dass Hyperledger Iroha sich als Blockchain Technologie für DIVA.EXCHANGE nicht eignet. DIVA.EXCHANGE plant als dezentrales Handelsnetzwerk Benutzern zugänglich zu sein, die nicht immer erreichbar sind. Dieses Verhalten der Benutzer führt bei Hyperledger Iroha allerdings zu langen Verzögerungen.

DIVA.EXCHANGE ist seit Q1/2021 an der Entwicklung einer eigenen Blockchain-Technologie, der "divachain". Die Sicherheit von "divachain" soll in einer weiteren Forschungsarbeit untersucht werden.

Abstract

We are investigating the DIVA.EXCHANGE project and try to identify possible security risks. DIVA.EXCHANGE is an open-source project, which aims to build a fully decentralized trading network using blockchain technology. The blockchain used is based on Hyperledger Iroha and is the primary subject of our research and our tests.

A test blockchain from DIVA.EXCHANGE is set up on a local network. Several tests are carried out within this test environment and the behavior of the blockchain is analyzed. The focus lies on the manipulation of the blockchain by dishonest participants. Other possible attacks on DIVA.EXCHANGE are also considered.

The most important results show that the blockchain based on Hyperledger Iroha does not adhere to the Byzantine fault tolerance of more than two thirds of honest participants. The results also show that the YAC consensus algorithm used by Hyperledger Iroha leads to long delays when creating blocks within the network, if network participants are not constantly available.

We conclude from the data that Hyperledger Iroha has clear security risks since Byzantine errors cannot be ruled out. This fact must be improved in order to guarantee that transactions of assets in the public context can take place without risk. We also conclude from our results that Hyperledger Iroha is not suitable as a blockchain technology for DIVA.EXCHANGE. DIVA.EXCHANGE plans to be accessible to users who are not always available to the network. However, this behavior of the users leads to long delays within Hyperledger Iroha.

DIVA.EXCHANGE has been developing its own blockchain technology, the "divachain", since Q1 / 2021. The security of "divachain" has to be examined in a further research project.

Vorwort

Die Themenwahl für eine Bachelorarbeit sollte man nicht auf die leichte Schulter nehmen. Vor allem wenn es eine Entscheidung ist, mit der ein Team leben muss, sollte sie gut durchdacht werden. Ein dementsprechend langer Prozess war es auch, sich zu einigen.

DIVA.EXCHANGE bot uns die Möglichkeit in eine spannende Technologie, die wir nur oberflächlich kannten, Einblick zu erhalten. Zusammen mit dem Team von DIVA.EXCHANGE hatten wir die Möglichkeit die Welt der Blockchains und Shared Ledgers mit einem sinnstiftenden Ziel unter die Lupe zu nehmen. Ein Open Source Projekt, um Banking zu dezentralisieren ist unserer Meinung nach eine interessante Entwicklung. Die Idee hat sowohl unsere Zustimmung gefunden als auch unser Interesse geweckt. Die Mischung aus einer sinnvollen Aufgabe, einer spannenden Technologie und einem Betreuer, den wir während dem Hochschulbetrieb bereits zu schätzen gelernt hatten, hat unsere Wahl erleichtert.

Während dem Voranschreiten der Arbeit erhielten wir aus erster Hand Einblick in die genutzten Technologien und Vorgehensweisen beim Aufbauen und Benutzen eines verteilten Speichers. Sowohl bei der Recherche als auch beim Testen gab es viel zu lernen. Wir beenden unsere Bachelorarbeit mit einem tiefen Verständnis zu Herausforderungen und Gegenmassnahmen bei Angriffen auf verteilte Speicher.

Wir möchten allen danken, die uns unterstützt und geholfen haben unsere Erkenntnisse zu Papier zu bringen.

Carolyn Bächler-Schenk und Konrad Bächler danken wir für den warmen Empfang bei DIVA.EXCHANGE, ihre Hilfsbereitschaft bei Problemen und ihren Input über die gesamte Arbeit.

Stephan Neuhaus danken wir für das Begleiten unserer Arbeit, seine Fähigkeit uns unermüdlich zu motivieren und seinen direkten Beitrag durch Korrektur und Verbesserungsvorschläge zur Methodik und technischen Aspekten.

Zu guter Letzt danken wir Marina Brägger für ihren Einsatz bei der gewissenhaften Rechtschreibprüfung und grammatischer Korrektur der ganzen Arbeit.

Inhaltsverzeichnis

Formular.....	1
Eigenständigkeitserklärung.....	1
Zusammenfassung	2
Abstract.....	3
Vorwort.....	4
Inhaltsverzeichnis.....	5
1 Einleitung	8
1.1 Ausgangslage.....	8
1.2 Aufgaben	9
1.3 Zielsetzung	9
1.4 Hypothesen.....	9
1.4.1 Abweisung von Blöcken ohne Konsens.....	9
1.4.2 Abweisung von nicht-standardgemässen Blöcken	10
1.4.3 Manipulationssicheres Quorum.....	10
1.4.4 Sicherheitsgerechter Gebrauch der API.....	10
1.4.5 Vollständige Anonymität.....	10
2 Theoretische Grundlage.....	11
2.1 DIVA.EXCHANGE	11
2.1.1 Diva	11
2.1.2 Hyperledger Iroha	12
2.2 Shared Ledger: Anforderung und Umsetzung	13
2.2.1 Shared Ledger	13
2.2.2 Anforderungen.....	13
2.2.3 Umsetzung mit einer Blockchain	14
2.3 Komponenten	15
2.3.1 Byzantinische Fehlertoleranz	15
2.3.2 Konsensalgorithmen	15
2.3.3 YAC	16
2.4 Funktionsweise Anhand einer Transaktion.....	17
2.5 Historische Aufarbeitung	19
2.6 Angriffe – Theorie und Beispiele.....	20
2.6.1 API Abuse	20
2.6.2 (D)DoS, 51 % Attack, Liveness Denial und Sybil Attack.....	21
2.6.3 Hash Collision	23

2.6.4	Replay Attack	23
2.6.5	Signature Forging	24
2.6.6	Supply Chain Attack	24
3	Methodik.....	25
3.1	Sicherheitskritische Aspekte aufdecken	25
3.1.1	"wird nicht untersucht" - Definition	25
3.1.2	Abuse-Cases	26
3.1.3	Data-Flow-Diagramm	28
3.1.4	Threat-Modelling	29
3.2	Teststand – Aufbau bis Auswertung	35
3.2.1	Erstellen und Anpassen der Knoten.....	35
3.2.2	Starten des Netzwerks	36
3.2.3	Ablauf der Tests	36
3.2.4	Auswertung der Tests	36
3.3	Tests – Beschreibung	37
3.3.1	Test T1 – Neue Einträge in die Blockchain verhindern	37
3.3.2	Test T2 – Blockchain manipulieren mithilfe Netzwerkangriff.....	39
3.3.3	Test T3 – Blockchain manipulieren mithilfe gewisser Mehrheit.....	41
3.3.4	Netzwerkgrößen für Tests.....	42
4	Resultate	43
4.1	Statische Analyse	43
4.1.1	S1 – Verwendete Verschlüsselung und deren Implementation	43
4.1.2	S2 – Zugriffsrechte auf lokale Schlüssel	46
4.1.3	S3 – Zugriffsrechte und Anpassung der lokalen Blockchain	46
4.2	Praktische Analyse – Teststand.....	47
4.2.1	Übersicht & Evaluation	47
4.3	Risikominimierung	58
4.3.1	R1 - Veraltete und unbekannte Verschlüsselungen.....	58
4.3.2	R2 - Sichere Passwörter und regelmässige Code Checks.....	58
5	Diskussion	59
5.1	Erkenntnisse.....	59
5.2	Falsifizieren der Hypothesen	60
5.2.1	Hypothese 1	60
5.2.2	Hypothese 2	60
5.2.3	Hypothese 3	60
5.2.4	Hypothese 4	61

5.2.5	Hypothese 5	61
5.3	Fazit.....	62
5.4	Empfehlungen an DIVA.EXCHANGE	63
5.4.1	Aspekt 1 – Quorum und Manipulationsschutz	63
5.4.2	Aspekt 2 – Auswahl des Abstimmungsorganisators	65
5.4.3	Aspekt 3 – Performance bei Abstimmungen	65
5.4.4	Aspekt 4 – Entfernung von Knoten	65
5.4.5	Aspekt 5 – Externe APIs und API Abuse	66
5.4.6	Aspekt 6 – DIVA.EXCHANGE Entwickler.....	66
6	Verzeichnisse	67
6.1	Literaturverzeichnis	67
6.2	Glossar.....	69
6.3	Abbildungsverzeichnis	71
7	Anhang	72
7.1	Projektmanagement	72
7.1.1	Aufgabenstellung	72
7.1.2	Zeitplan	73
7.1.3	Meeting Notizen	75
7.1.4	Protokolle.....	76
7.2	Weiteres.....	83
7.2.1	Tests Code	83
7.2.2	Resultate – Tabellen.....	85
7.2.3	Kosten für 51 % Angriff auf Iroha Blockchain	94
7.2.4	Helperskripte	95

1 Einleitung

1.1 Ausgangslage

In dem vorliegenden Forschungsbericht wird die Sicherheit des vollständig verteilten, Blockchain-basierten, Open Source Handelsnetzwerkes DIVA.EXCHANGE [1] analysiert. Dazu wird insbesondere die Blockchain-Ebene "Hyperledger Iroha" [2] untersucht. Es soll festgestellt werden, inwiefern sicherheitskritische Mängel in Hyperledger Iroha oder zugehörigen Komponenten vorhanden sind, die es Angreifern ermöglichen, Einfluss auf DIVA.EXCHANGE und seine Benutzer zu nehmen.

DIVA.EXCHANGE ist ein Projekt, das ein Handelsnetzwerk aufbaut, welches eine vollständig verteilte Datenablage Privatsphäre-schützend ermöglichen will. Die Sicherheit der Benutzerdaten und die Anonymität der Benutzer selbst sind dabei von oberster Priorität. Aus diesem Grund erfolgt die Kommunikation bei DIVA.EXCHANGE über I2P. Obwohl es sich bei DIVA.EXCHANGE um ein privates Blockchain-Netzwerk handelt, soll keine Registration mit persönlichen Daten erfolgen. Sämtlicher Code zum Projekt ist öffentlich und als Open Source Software einsehbar [1].

Wie sicher ist ein solches System, welches auf einer Blockchain basiert? Blockchain Systeme können, wie alle Software-Systeme, sicherheitskritische Mängel aufweisen. Es gibt bereits einige Beispiele für fehlerhaftes Verhalten in der Vergangenheit, wie Transaktionen, die aufgrund von arithmetischen Überläufen falsch ausgeführt wurden [3]. Aber auch bösartige Angriffe, um Wertmittel eines Benutzers zu stehlen oder zu vernichten, sind bereits in der Vergangenheit ausgeführt worden [4].

Die Sicherheit von Transaktion und Besitz im digitalen "Wallet" beruht auf den Eigenheiten der Blockchain-Technologien. Eine Blockchain ist eine kontinuierlich wachsende und erweiterbare Kette von Datenätzen, die auch «Blöcke» genannt werden. Dabei beinhaltet jeder Block einen kryptographischen Hash, der zum vorherigen Block gehört. Alle beteiligten haben auf ihrem eigenen System eine Abbildung der Kette, die auf diese Weise entsteht. Damit nun sichergestellt wird, dass alle dieselbe Kette betrachten und niemand eine Manipulation oder einen Fehler in seinem System mitbringt, gibt es sogenannte Validatoren, die beim Hinzufügen eines Blocks bestätigen, dass die dieser und die Kette ihre Richtigkeit haben [5].

1.2 Aufgaben

Die Aufgaben bilden die Ausgangslage der Zielsetzung und der anschliessenden Untersuchung. Die folgenden Aspekte sind von DIVA.EXCHANGE im Auftrag definiert worden:

- Aufzeigen der Funktionsweise und der Protokolle von DIVA.EXCHANGE und die Blockchain-Ebene Hyperledger Iroha
- Abstraktion und Formalisierung von Angriffsszenarien
- Erstellen eines Teststands mit definierbaren (d.h. konfigurierbaren) Angriffs-Szenarien und definierten Testerwartungen
- Ausführen des Teststands mit den definierten Angriffs-Szenarien, beispielsweise ein Replay-Angriff
- Zusammenfassung, Dokumentation und Visualisierung der Test-Resultate

1.3 Zielsetzung

Die Hypothesen des folgenden Abschnittes sollen untersucht und die Resultate dokumentiert werden. Für ein besseres Verständnis der verwendeten und aufgeführten Technologie wird im nächsten Kapitel auf die theoretischen Grundlagen eingegangen sowie die Methodik, mit der die Resultate erzeugt werden, erläutert. Damit soll das Vertrauen in die Sicherheit von DIVA.EXCHANGE erhöht, oder alternativ ein Wechsel oder Überarbeiten der Blockchain-Technologie begründet werden.

1.4 Hypothesen

Gemeinsam mit DIVA.EXCHANGE und Dr. Neuhaus sind die folgenden Hypothesen aufgestellt worden. Mithilfe der kommenden Kapiteln werden diese genauer untersucht. Dazu wird im folgenden Kapitel als erstes die Theorie einer Blockchain im Allgemeinen aufgezeigt und die Theorie von konkreten Angriffen genauer beschrieben. Folgend werden mögliche sicherheitskritische Aspekte analysiert. Anschliessend werden jene Aspekte, die echte Sicherheitslücken darstellen könnten, nochmals genauer untersucht. Schlussendlich werden die Auswirkungen von diesen Sicherheitslücken diskutiert oder aufgezeigt, wieso es doch keine Sicherheitslücken sind.

1.4.1 Abweisung von Blöcken ohne Konsens

Der vollständig verteilte Handel bei DIVA.EXCHANGE findet über eine Blockchain statt. Aus Sicht des Handelsnetzwerks ist diese Blockchain ein öffentlicher und im Nachhinein nicht änderbarer Datenspeicher. Damit neue Daten aufgenommen werden können, müssen diese in einen sogenannten Block eingebettet werden und die Teilnehmer des Netzwerks, also die Knoten, müssen diesen Block mit einer gewissen Mehrheit akzeptieren. Diese Mehrheit nennt sich "Konsens". Es darf nicht dazu kommen, dass Blöcke, die nicht den nötigen Konsens aufweisen, in die Blockchain aufgenommen werden. Eine ausführlichere Erklärung dieses Konsenses finden sich in Kapitel 2.1.

HYPOTHESE 1: Die Blockchain-Ebene weist alle Blöcke ab, bei denen die Teilnehmer keinen Konsens finden.

1.4.2 Abweisung von nicht-standardgemässen Blöcken

Die Daten, die ein neuer Block enthält, müssen gewisse Vorgaben erfüllen. Sollten gewisse Daten in einem Block nicht die von DIVA.EXCHANGE gestellten Vorgaben erfüllen, soll der enthaltende Block nicht in die Blockchain aufgenommen werden.

HYPOTHESE 2: Daten innerhalb eines Blockes sind von den betroffenen Personen autorisiert, sind stets authentisch und konform mit den Vorgaben von DIVA.EXCHANGE.

1.4.3 Manipulationssicheres Quorum

Alle Knoten von DIVA.EXCHANGE sind automatisch bei Abstimmungen dabei, wenn neue Blöcke in die Blockchain aufgenommen werden sollen. Das Aufnehmen von neuen Blöcken stellt dazu keine direkten sicherheitskritischen Probleme dar. Jedoch müssen die Knoten, die nicht mehr aktiv sind, in sicherer Art und Weise behandelt werden. Werden sie langsam oder nie vom Netzwerk entfernt, wird es immer schwieriger, einen Konsens zu erreichen. Werden sie zu schnell entfernt, zum Beispiel von anderen Knoten im Netzwerk, kann das Quorum, also die Mindestanzahl der Knoten, die einer Aufnahme zustimmen müssen, kurzzeitig reduziert werden. Somit könnte auch mit einer Minderheit neue Knoten aufgenommen werden.

HYPOTHESE 3: Das Quorum ist resistent gegen böswillige Manipulationen.

1.4.4 Sicherheitsgerechter Gebrauch der API

Für das verteilte Speichern von Daten verwendet DIVA.EXCHANGE eine Open Source Blockchain Technologie (Iroha, Stand 12.04.2021). Auch für den Austausch zwischen den Knoten, wird eine Open Source Kommunikationstechnologie verwendet (I2P, Stand 12.04.2021). Diese beiden Technologien bieten ebenfalls eine API an, ebenso wie die selbst entwickelte API von DIVA.EXCHANGE. Es dürfen keine falschen Annahmen zur Funktionsweise dieser APIs getroffen werden.

HYPOTHESE 4: DIVA.EXCHANGE verwendet die eigene und externe APIs in sicherer Art und Weise.

1.4.5 Vollständige Anonymität

Die privaten Daten eines Knoten sind nur für den autorisierten Benutzer dieses Knotens ersichtlich. Die lokale Kopie der Blockchain kann nur bei Eintreffen eines neuen Blocks, der die Zustimmungsrate erfüllt, oder vom Benutzer selbst angepasst werden.

HYPOTHESE 5: Die Knoten sind vollständig anonym und können nicht von anderen Knoten im Netzwerk manipuliert werden.

Der Versuch der Falsifizierung dieser Hypothesen wird in Kapitel 4 aufgezeigt und die Auswirkungen in Kapitel 5 geschildert.

2 Theoretische Grundlage

2.1 DIVA.EXCHANGE

Um Angriffspunkte im Projekt DIVA.EXCHANGE aufzuspüren und zu nutzen, ist es wichtig zu verstehen was DIVA.EXCHANGE ist.

2.1.1 Diva

Das Ziel von DIVA.EXCHANGE ist ein vollständig verteiltes Banksystem aufzubauen, um freie Banking Technologie für alle anzubieten. "Die Idee von DIVA.EXCHANGE "Jeder ist seine eigene Bank", führt theoretisch zur vollständigen Selbstbestimmung über die Allokation eigener digitalen Werte. Eine Bank ist einfach ein Vermittler, z.B. für Zahlungen, für den Handel oder für Kredite. Wenn also jeder seine eigene Bank wäre, wäre das Netzwerk der Markt. [6]

DIVA.EXCHANGE stellt einen vollständig verteilten Datenspeicher, einen "Ledger" bereit, um dieses vollständig verteilte System aufzubauen. Dieser Ledger basiert auf der Blockchain-Technologie, welche (Stand 12.04.2021) durch Hyperledger Iroha implementiert wird.

Ein vollständig verteilter Ledger, auch Shared oder Distributed Ledger, ist eine Art der Speicherung von Daten. Die Grundidee ist es, Daten anstatt an einem zentralen Ort zu speichern, auf viele oder auch alle Knoten eines Netzwerkes zu verteilen. Mehr dazu folgt im nächsten Kapitel "2.2 Shared Ledger".

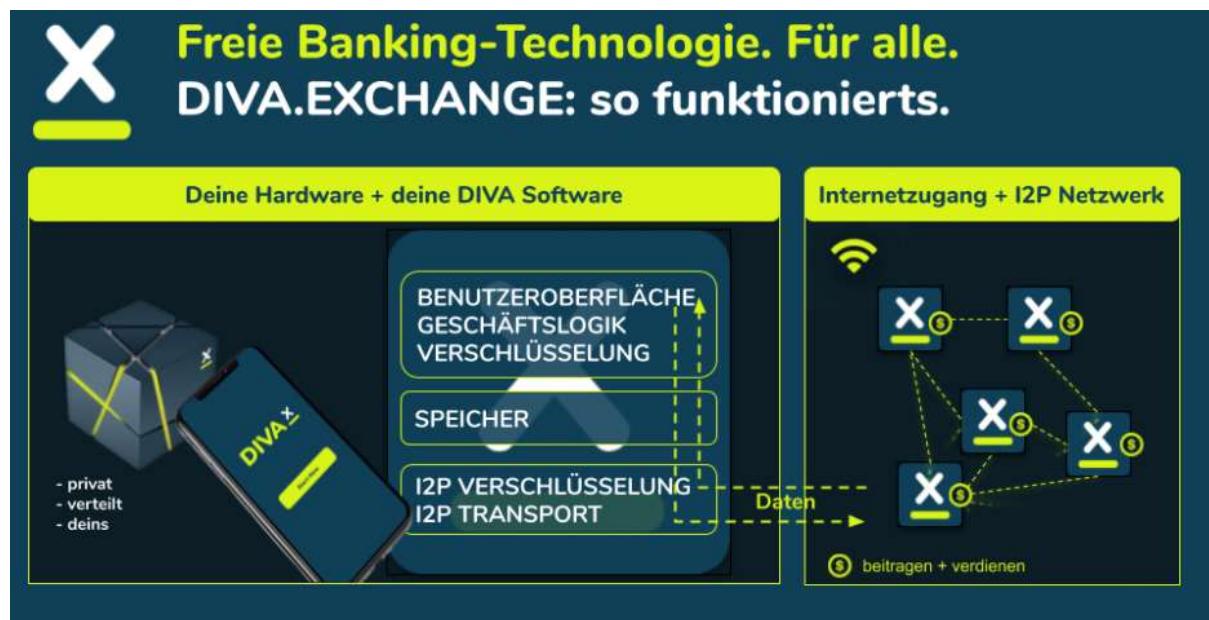


Abbildung 1: DIVA.EXCHANGE [6]

Die Kommunikation bei DIVA.EXCHANGE erfolgt wie in Abbildung (1) dargestellt über ein I2P Netzwerk. Daten werden verschlüsselt transportiert und lokal in einer Blockchain gespeichert. Die Umsetzung des Ledgers als Blockchain ermöglicht Transparenz über die ablaufenden Funktionen. Ein Layer darüber ist die Benutzeroberfläche von DIVA.EXCHANGE [6].

2.1.2 Hyperledger Iroha

Hyperledger Iroha ist eine eigenständige Implementation eines vollständig verteilten Ledgers mit Identitätsmanagement durch die Blockchain Technologie, die von Hyperledger entwickelt und von der Linux Foundation unterstützt wird. Hyperledger Iroha ist in C++ geschrieben und stellt eine Implementierung der Blockchain Technologie als Open Source Projekt zur Verfügung. Hyperledger Iroha ist dabei als eine flexibel nutzbare Bibliothek zu verstehen, mit der eine selbst konfigurierbare Blockchain aufgesetzt werden kann [2].

Bei Hyperleger Iroha handelt es sich um eine private Blockchain. Eine interne oder native Kryptowährung, die für die Interaktion mit dem System erforderlich ist, gibt es auf Iroha nicht. Die üblichen Blockchains, welche Coins und Tokens vertreiben, sind jedem frei zugänglich, der mit ihnen handeln will und haben keine speziellen Berechtigungen für Benutzer vorgesehen. Hyperledger Iroha ermöglicht es hingegen, Shared oder Distributed Ledgers in verschiedenen Größen für verschiedene Netzwerke aufzubauen und nur bestimmten Benutzern bestimmte Berechtigungen zu erteilen [7].

Ein Account bei Hyperledger Iroha kann als eine Entität gesehen werden, die ermächtigt ist, eine gewisse Menge an Aktionen vorzunehmen. Jeder Account hat eine Reihe von Rollen (ein Account kann auch keine Rolle haben). Dabei ist eine Rolle wiederum nur eine Sammlung von Berechtigungen innerhalb des Netzwerks, respektive innerhalb der Blockchain [8].

Die Manipulation von Accounts und digitalen Vermögenswerten erfolgen über die Iroha API. Dazu werden Kommandos und Queries benutzt. Die ausgeführten Kommandos sind atomare Status verändernde Aktionen, welche in einzelnen Transaktionen eingebunden werden können. Queries hingegen erlauben es Iroha Benutzern Informationen über eigenen Transaktionen und das eigene Vermögen aus der Blockchain zu erhalten [8].

2.2 Shared Ledger: Anforderung und Umsetzung

2.2.1 Shared Ledger

Bei einem Shared oder Distributed Ledger besitzt jeder Knoten eine exakte Kopie dieses Ledgers, also aller Daten. Diese Struktur hat zwei grundlegende Aufgaben. Zum einen soll durch das Verteilen der Daten sichergestellt werden, dass keine Daten verloren gehen. Zum anderen müssen sich die Teilnehmer durch die Struktur der verteilten Daten nicht gegenseitig vertrauen. Ein Shared Ledger stellt die korrekte Verteilung und Prüfung der eigenen und fremden Transaktionen sicher. Shared Ledgers können in der Informatik als Speicher oder Kommunikationsmittel fungieren. In der Praxis werden sie hauptsächlich als eine Art Kassenbuch genutzt, um Transaktionen von Wertmitteln wie zum Beispiel Bitcoin oder Ethereum dezentral zu dokumentieren. Die Hauptaufgabe ist es, Wertmittel möglichst sicher ohne zentrale Prüfstelle, wie zum Beispiel eine Bank, auszuführen. Um ein Shared Ledger ohne zentrale Prüfstelle zu implementieren, müssen Daten möglichst automatisiert erfasst, geprüft und geteilt werden. Dieser Vorgang muss transparent sein und ohne Administrator oder menschlichen Eingriff stattfinden, damit auch sich unbekannte Teilnehmer dem Vorgang vollständig vertrauen. [9]

2.2.2 Anforderungen

Die Anforderungen an ein Distributed Ledger sind nach der International Telecommunication Union [10] folgende:

- **Decentralization:** Ein Shared Ledger muss auf Kooperation aller im Netzwerk vorhandener über peer-to-peer (P2P) verbundener Knoten aufbauen. Jeder Knoten muss eine lokale Kopie des Shared Ledgers besitzen.
- **Proof of Ownership:** Ein Shared Ledger ist ein Datenspeicher, der Daten final und unveränderbar ablegt. Es geht dabei sowohl um Wertmittel als auch Transaktionen. Das Shared Ledger muss zu jedem Zeitpunkt festlegen wer was besitzt.
- **Data validity:** Jeder Knoten im Netzwerk muss die Arbeit anderer Knoten überprüfen können, da es keine vertrauenswürdigen zentralen Knoten gibt. Jede Nachricht muss verifiziert werden, um als valide zu gelten.
- **Tamper-resistance:** Sobald Daten im Distributed Ledger gespeichert sind, müssen diese unveränderbar sein.
- **Auditability:** Die Überprüfung von Transaktionen und Daten muss in einem öffentlichen Shared Ledger von allen teilnehmenden Knoten ausgeführt werden können. In privaten Shared Ledgers kann dieser Punkt entfallen, wenn es in diesen Ledger privilegierte Benutzer hat.
- **Fariness:** Auch wenn es zwischen teilnehmenden Knoten Unterschiede bei der Stabilität ihrer Verbindung und ihrer Rechenleistung gibt, müssen sie alle möglichst gleichbehandelt werden, um sicherzustellen, dass die Validität des ganzen Shared Ledgers von möglichst vielen Parteien sichergestellt wird.
- **Stability:** Durch die dezentrale Natur des Shared Ledgers kann es dazu kommen, dass verschiedene abzweigende Versionen auf verschiedenen Knoten entstehen. Diese Konflikte müssen automatisch und mit klaren Regeln und Vorgehen gelöst werden, so dass immer nur eine wahre Version des Shared Ledgers existiert.
- **Governance:** Durch die dezentrale Natur des Shared Ledgers muss sichergestellt werden, dass es auch ohne zentralen Besitzer immer korrekt funktioniert.

2.2.3 Umsetzung mit einer Blockchain

Um ein Shared Ledger zu implementieren, wird die Blockchain Technologie genutzt. Bei einer Blockchain handelt es sich um ein System, bei dem Datensätze zusammen in Blöcken abgespeichert werden. Diese Blöcke werden aneinandergereiht. Da jeder Block den Hashwert des Inhalts des vorherigen Blockes beinhaltet, kann kein Block verändert werden, ohne alle nachfolgenden Blöcke zu ändern. Das stellt die Struktur des Shared Ledgers sicher. Durch die Funktionsweise einer Blockchain als Datenspeicher werden Proof of Ownership und Tamper-resistance erfüllt. Die Decentralization entsteht dadurch, dass jeder Benutzer eine eigene vollständige Version der Blockchain lokal besitzt. Transaktionen, neue Blöcke und Validationen werden kommuniziert. Eine Version des Shared Ledgers besitzt jeder Benutzer lokal.

Um Data validity einzuhalten, prüfen immer mehrere Knoten gleichzeitig Transaktionen und erstellen neue Blöcke. Durch die Mehrfachprüfung kontrollieren sich die Knoten im Netzwerk gegenseitig. Nun kann es dazu kommen, dass durch Fehler oder bösartige Eingriffe kurzzeitig mehrere abzweigende Varianten der Blockchain entstehen. Damit alle Knoten dieselbe Version der Blockchain benutzen, müssen sie sich auf eine wahre und aktuelle Version einigen. Dieser Vorgang nennt sich Konsensverfahren und kann sich von Implementation zu Implementation der Blockchain unterscheiden. Das Konsensverfahren stellt Stability und je nach Implementation Fairness sicher.

Damit ein solches Distributed Ledger auf einer Blockchain funktioniert, muss es möglichst selbstständig arbeiten. Jeder legitime Benutzer im Netzwerk hat Interesse daran, dass Transaktionen korrekt ablaufen. Die Benutzer prüfen Transaktionen und sich gegenseitig. Sie stellen damit selbst die Rechenleistung für die Operationen, die auf der Blockchain ausgeführt werden müssen, bereit. Fehler oder bösartige Angriffe auf die Blockchain sollen vom Konsensverfahren durch eine Mehrheit abgewehrt werden. Damit dies allerdings immer gelingt muss die byzantinische Fehlertoleranz eingehalten werden. Diese Punkte stellen Governance sicher.

Audability wird von Blockchain zu Blockchain anders gehandhabt.

2.3 Komponenten

2.3.1 Byzantinische Fehlertoleranz

Der byzantinische Fehler beschreibt in der Informatik einen Zustand in einem Computersystem, indem einzelne Komponenten Fehler aufweisen könnten und keine sicheren Informationen verfügbar sind, ob in einer Komponente wirklich ein Fehler aufgetreten ist. Die Herkunft des Begriffes geht auf die Problemstellung zurück, in welcher sich mehrere Komponenten auf ein strategisches Vorgehen einigen müssen, um einen Systemausfall zu verhindern. Einige Komponenten sind dabei aber nicht vertrauenswürdig. In der ursprünglichen Idee bezieht man sich dabei auf byzantinische Generäle als Komponenten, welche sich für eine Strategie für ihre Armeen aus ihren jeweiligen Hauptquartieren heraus entscheiden müssen, ohne sicher zu sein, was die anderen genau planen [11].

Wenn ein byzantinischer Fehler auftritt, kann ein Beobachter bei Betrachtung einzelner Komponenten nicht immer ausmachen, ob ein Fehler stattfindet und welche Komponente ihn verursacht. Gleichzeitig ist es für andere Komponenten im System schwierig, einer Komponente einen Fehler vorzuwerfen, da erst ein Konsens darüber gefunden werden muss. Im Beispiel der Blockchain werden die einzelnen Akteure betrachtet, sowie die Implementationen der Blockchain im verteilten System. Ein Knoten innerhalb der Blockchain könnte fehlerhafte Informationen senden. Andere Knoten können außerdem nicht allein einen fehlerhaften Knoten korrigieren oder aus dem System werfen. Daher braucht es eine Art System zum Erreichen eines Konsenses von allen Knoten, welcher Knoten im System fehlerhafte Informationen herausgibt [12].

Ein Blockchain System muss automatisiert, ohne Eingriff von aussen, korrekt funktionieren. Um das zu gewährleisten, müssen byzantinische Fehler ausgeschlossen werden. Castro und Liskov konnten in ihrer Arbeit "Practical Byzantine Fault Tolerance and Proactive Recovery" 2002 zeigen, dass byzantinische Fehler zwischen Komponenten, wie in einem Blockchain System, nur dann verhindert werden können, wenn die Anzahl fehlerhafter oder unehrlicher Knoten kleiner als ein Drittel der gesamten Anzahl Knoten im Netzwerk ist [13].

2.3.2 Konsensalgorithmen

Konsensalgorithmen sind ein zentraler Bestandteil von Blockchain-Systemen. Es handelt sich um ein Vorgehen, bei dem alle beteiligten Knoten zusammen eine Vereinbarung darüber treffen, welche Version der Blockchain und damit des Distributed Ledgers aktuell ist. Damit wird das Vertrauen zwischen den sich sonst fremden Benutzern überflüssig. Der Konsensalgorithmus garantiert, dass mit jedem neu hinzugefügten Block Sicherheit darüber herrscht, dass es nur eine allgemeingültige Version der Blockchain gibt und kein byzantinischer Fehler auftreten kann [14].

Es gibt mehrere Arten der Konsensalgorithmen. Zu den relevantesten gehört Proof of Stake. (PoS) [12].

Proof of Stake Algorithmen sind Konsensverfahren, die mithilfe eines gewichtetes Zufallsverfahren die Entscheidung darüber fällen, wer einen neuen Block erstellen soll. Zufällig gewählte Knoten stimmen untereinander ab welche Version der Blockchain die aktuelle ist. Die Abstimmung ist dann für das ganze System bindend. Die Gewichtung jedes einzelnen Knotens basiert dabei auf seiner Teilnahme-dauer der Blockchain und dem Vermögen des Teilnehmers innerhalb der Blockchain, also seinem Stake. Proof of Stake Algorithmen sind daher unsicher gegenüber Teilnehmern, die mehr als die Hälfte des gesamten Stakes im Netzwerk auf sich vereinen [15].

Eine Weiterentwicklung dieses Systems sind Delegated Proof of Stake Algorithmen. Bei Delegated Proof of Stake Algorithmen wählen alle am Netzwerk teilnehmenden Knoten eine Gruppe von vertrauenswürdigen Knoten. Diese gewählte Gruppe erhält das Recht, nacheinander neue Blöcke zu erstellen. Jedes Mitglied der gewählten Gruppe erhält als Belohnung für die Erstellung eines Blocks eine Transaktionsgebühr ausbezahlt. Nicht jeder Knoten hat dasselbe Gewicht bei der Wahl. Der Stimmanteil, den ein Knoten besitzt, basiert auf der Grösse seines Stakes. Dieses Verfahren soll sicherstellen, dass vertrauenswürdige und für das Netzwerk positive Knoten Blöcke erstellen. Stakeholder wählen in ihrem Interesse Knoten, denen sie vertrauen. Je grösser der eigenen Stake ist, desto mehr Interesse hat man, dass die Blockchain einwandfrei funktioniert. Und Knoten haben auch durch Bezahlung der Blockerstellung einen Grund, sich wählen lassen zu wollen [16].

2.3.3 YAC

Hyperledger Iroha verwendet als Konsens-Verfahren den YAC-Algorithmus. YAC steht für "Yet Another Distributed Consensus Algorithm", und ist ein Proof of Stake Verfahren.

YAC unterscheidet zwischen Clients und Peers. Clients sind alle an der Blockchain teilnehmenden Benutzer, die einen Public-Key innerhalb des Netzwerkes besitzen. Ein Peer hingegen ist ein Netzwerkteilnehmer, der Sonderrechte und Pflichten hat. Peers sind verantwortlich für Validation von Transaktionen, das Erstellen von Blocks und das Abstimmen über neue Blocks. Bei YAC besitzen nur Peers eine vollständige Historie der gesamten Blockchain.

Peers fungieren als Validatoren, die "Proposal" enthalten, in welchen mehrere Transaktionen überprüft werden müssen. Jeder Peer hat Zugriff auf Werkzeuge, die es ihm ermöglichen, Transaktionen für gültig oder ungültig zu erklären. Gültige Transaktionen werden zu neuen Blöcken zusammengelegt und gehashed. Mit dem entstandenen Hashwert wird eine Permutation von Peers berechnet. Dazu wird bei YAC die "Order Funktion" genutzt. Der Vorschlag des neuen Blocks wird dann zum nächsten Peer innerhalb dieser Validationskette von Peer gesendet, nachdem der Block unterschrieben wurde. Der nächste Peer überprüft ebenfalls alle Transaktionen innerhalb des Blocks, berechnet den Hash und wenn alle Daten übereinstimmen, erhält er dieselbe Permutation wie der Peer vor ihm. Sobald die benötigte Mehrheit von mehr als zwei Dritteln der Peers erreicht ist, wird der Block an die Blockchain gehängt und eine Commit-Nachricht an alle Peers gesendet [17].

2.4 Funktionsweise Anhand einer Transaktion

Um die Funktionsweise eines Shared Ledger und dessen Umsetzung als Blockchain genauer zu erklären, wird hier der Ablauf einer Transaktion auf einer Blockchain gezeigt.

Ein Benutzer, der wünscht, eine Transaktion zu tätigen, muss zuerst über die entsprechende Software/das Wallet verfügen. In unserem Falle stellt dieses die Benutzeroberfläche von DIVA.EXCHANGE zur Verfügung. Die Blockchain-Technologie funktioniert immer nach denselben Prinzipien, unterscheiden sich aber je nach Implementierung. Eine Transaktion läuft aus Sicht der Blockchain folgendermassen ab:

Als Erstes erzeugt der Sender ein Public-Key, Private-Key Schlüsselpaar. Der Private-Key wird verwendet, um die Transaktion und die damit versendeten Daten zu signieren. Der Public Key ermöglicht später allen Benutzern zu verifizieren, dass die Signatur des Senders, und damit die gesamte Transaktion, authentisch ist. Um nun zu senden, wird ebenfalls noch eine Adresse benötigt. [18].

Die Transaktion wird nach festgelegten Richtlinien erstellt und muss demnach einige sensible Daten enthalten. Zum Beispiel Zieladresse, Betrag, und auch Referenzen auf alle bisher erfolgreichen Transaktionen des Senders. Diese Transaktion wird mit Hilfe des Private Keys des Senders signiert und dann verschlüsselt zusammen mit dem Public Key an den Empfänger gesendet. Der Sendevorgang ist in Abbildung (2) aufgezeigt.

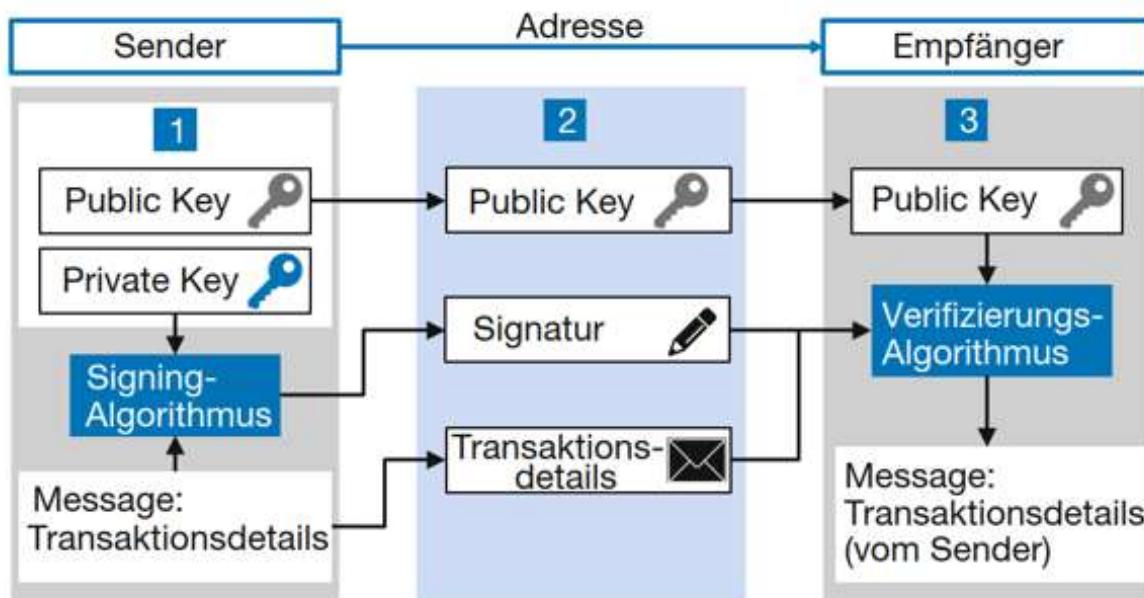


Abbildung 2: Sendevorgang [18]

Zuletzt kann der Empfänger nun mit Hilfe des Public Keys verifizieren, dass die Sendung tatsächlich vom Sender stammt und er über die benötigte Menge an Währung verfügt.

Die Authentizität dieser Transaktion wird dezentral geprüft. Positiv geprüfte Transaktionen werden zu Blöcken zusammengefasst. Jeder Block beinhaltet Transaktionen, Informationen und eine Referenz auf den vorherigen Block. Durch die Referenz auf den jeweilig vorherigen Block entsteht durch die Verkettung der Blöcke die Blockchain selbst.

Der Inhalt eines Blocks wird anschliessend gehashed. Dieser Hashwert wird hier als "Root" bezeichnet. In Abbildung (3) ist eine vereinfachte Darstellung einzelner Blöcke innerhalb der Blockchain zu sehen.

Die Root wird im Header des Blocks zusammen mit dem Hash des vorherigen Blocks abgespeichert.

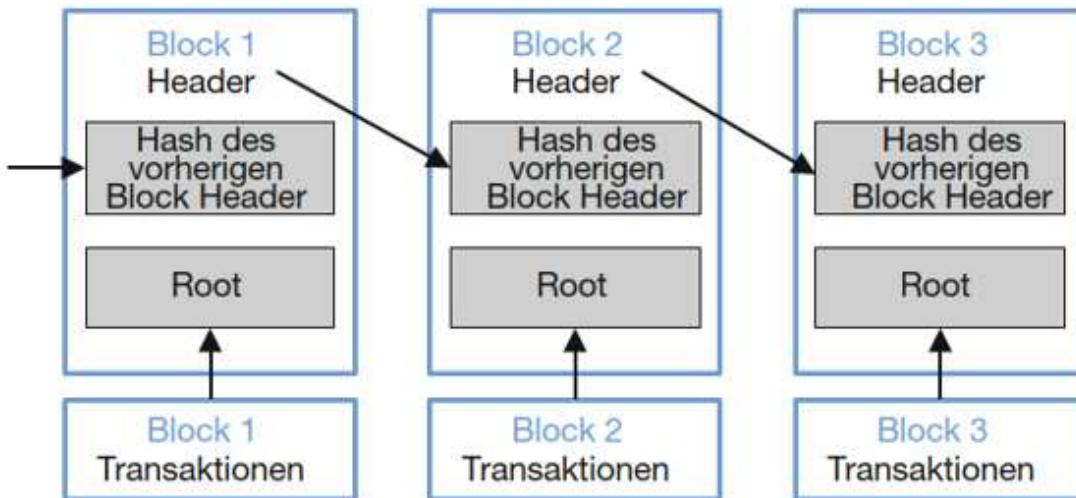


Abbildung 3: Blocks [18]

Sobald ein neuer Block fertiggestellt ist, wird seine Richtigkeit überprüft. Falls der neue Block Freigabe erhält, wird er zur Blockchain hinzugefügt. Dadurch, dass mehrere Knoten gleichzeitig Transaktionen prüfen und neue Blöcke erstellen, kann es dazu kommen, dass kurzzeitig mehrere abzweigende Varianten der Blockchain entstehen. Damit alle Knoten dieselbe Version der Blockchain benutzen, müssen sie sich per Konsensverfahren auf eine wahre und aktuelle Version einigen. Dieser Vorgang kann sich von Implementation zu Implementation der Blockchain unterscheiden.

Die Sicherheit des Systems gegenüber Fehlern und bösartigen Manipulationen entsteht nun dadurch, dass kein Block und keine Transaktion abgeändert werden kann, ohne alle darauffolgenden Blöcke zu korrumpieren. Sollte auch nur ein Wert in einem Block verändert oder beschädigt werden, stimmt der Hash nicht mehr mit dem des folgenden Knoten überein. Dadurch stimmt auch der Hash des nächsten Blocks mit dem des übernächsten Blocks nicht mehr überein, und so weiter. Der erste sich unterscheidende Block wird dann automatisch von der Mehrheit in der Blockchain aussortiert und die tatsächlich authentische Transaktion, respektive der tatsächlich authentische Block, wird an seine Stelle eingefügt.

Das Konzept der Blockchain verhindert so, dass Manipulationen an Transaktionen vorgenommen werden können. Ein Angreifer müsste über eine grosse Menge der Rechenleistung oder des Stakes des gesamten Netzwerks verfügen, um Transaktionen manipulieren zu können [19]. Denn nur so ist er im Stande, unautorisiert Blöcke an die Blockchain zu knüpfen.

Die Blockchain an sich beinhaltet alle Transaktion, die jemals über sie stattgefunden haben, so wie auch die Informationen, wer wieviel an Wertmitteln besitzt. Damit ist eine vollständige Historie aller jemals stattgefundenen Änderungen jederzeit verfügbar. Die Privatsphäre eines jeden Teilnehmenden ist trotzdem sichergestellt, da genaue Informationen zu Transaktionen nur mit den Privat Keys der jeweils beteiligten Parteien entschlüsselt werden können [18].

2.5 Historische Aufarbeitung

Die Idee der Verkettung von einzelnen Blöcken unter kryptografischen Sicherheitsmechanismen gibt es schon seit den 1990ern. Bereits 1991 wurde von Stuart Haber und W.Scott Stornetta die grundlegende Idee beschrieben [20].

1998 wird von Nick Szabo an einem Mechanismus für eine digitale dezentralisierte Währung gearbeitet. Er nennt sie "Bit Gold" [21].

Im Jahr 2000 veröffentlichte Stefan Konst seinen Bericht "Sichere Log-Dateien auf Grundlage kryptographische verketteter Einträge." [20]. Darin wird die Theorie für kryptografische Verkettung erarbeitet. Konst stellt fest, dass es verschiedenen Prinzipien gibt, nach denen die Sicherheit verketteter Blöcke herzustellen ist. Er geht dabei von Daten innerhalb einer Log-Datei aus. In seiner Zusammenfassung kommt Konst zum Schluss, dass die Verschlüsselung sich auch "[...] mittels Public-Key-Kryptosystemen realisieren lässt." [20]

Die Blockchain-Technologie als verteiltes Datenbankmanagementsystem entstand 2008 und wurde als integraler Bestandteil der digitalen Währung Bitcoin erarbeitet. Der Erschaffer des Bitcoins, bekannt als Satoshi Nakamoto, beschrieb die Theorie der Blockchain in seinem Paper Bitcoin: A Peer-to-Peer Electronic Cash System [22].

2.6 Angriffe – Theorie und Beispiele

In den folgenden Kapiteln wird zum Teil auf die folgenden Angriffe verwiesen. Um diese Angriffe nicht bei jeder Erwähnung zu erläutern, werden sie in diesem Abschnitt aufgelistet. Für jeden dieser Angriffe wird die Theorie und Funktionsweise dahinter aufgezeigt, darauf ein Beispiel oder eine Anwendung genannt, und abschliessend die generell akzeptierten Massnahmen gegen diese Angriffe aufgezählt.

2.6.1 API Abuse

Im Falle einer API Abuse trifft der Entwickler falsche Annahmen zur Funktionsweise oder Sicherheit einer API-Funktion. Im fertigen Produkt kann diese falsche Annahme dann von Angreifern ausgenutzt werden.

Beispielsweise erstellt der Entwickler eine Funktion für den Admin, beliebige Blöcke der Blockchain zu löschen. Der Zweck dahinter könnte sein, um Blöcke zu löschen, die gefälscht oder unautorisiert entstanden sind. Ob das sinnvoll ist, wird hier nicht weiter diskutiert. Nun könnte der Entwickler allerdings vergessen haben, dass der Admin des Unternehmens authentifiziert sein muss, um ein Block zu löschen. Die falsche Annahme des Entwicklers führt zum Fehlen einer Authentifizierung, was dazu führt, dass ein Angreifer, der natürlich nicht authentifiziert ist, beliebige Blöcke löschen kann.

Solche Schwachstellen können mit ordentlich geführtem Secure Development Lifecycle, ausführlichen Security Requirements und regelmässigen Code Reviews verhindert werden. Auch im Nachhinein können solche Schwachstellen mit Penetration Tests oder Rückmeldung von ethischen Hackern aufgedeckt werden, allerdings sind so die Kosten meist höher, es ist unmöglich zu implementieren, oder die Schwachstelle ist schon ausgenutzt worden [23].

2.6.2 (D)DoS, 51 % Attack, Liveness Denial und Sybil Attack

Unter (D)DoS versteht sich ein "(Distributed) Denial of Service", also einen Angriff, der die Nutzung eines Dienstes für legitime Benutzer vorübergehend verhindert. Dieser Angriff kann einem aber auch von mehreren Orten ausgehen, um erfolgreicher beziehungsweise schwieriger abzuwehren zu sein [24].

Meist wird ein Server mit so vielen Anfragen bombardiert, dass dieser unter der Last nicht mehr arbeiten kann. Weil DIVA.EXCHANGE und Iroha vollständig verteilt aufgebaut sind, gibt es auch keinen zentralen Server und kein geeignetes Ziel für herkömmliche (D)DoS Angriffe. Gegen einen einzelnen Knoten, also einen DIVA.EXCHANGE Benutzer, schützt das I2P Protokoll nur teilweise [25]. Aber weil die Knoten in I2P anonym sind, kann ein solcher Angriff nur gegen einen zufälligen I2P Knoten sein. Somit ist nicht einmal garantiert, dass der Angriff einen DIVA.EXCHANGE Benutzer trifft.

Allerdings ist bei Iroha zu beachten, dass zwei Drittel der Knoten aktiv einen neuen Block genehmigen müssen. Das heisst, sollte ein Drittel der Knoten absichtlich keine Antwort auf neu zu erstellende Blöcke geben, können keine neuen Daten in die Blockchain aufgenommen werden. Dies nennt sich auch "Liveness Denial" [26]. Ab zwei Drittel "börsartiger" Knoten sind nach der byzantinischen Fehlertoleranz sogar Angriffe möglich, bei denen beliebige Blöcke in die Blockchain aufgenommen werden.

Bei Proof of Work Blockchains, wie zum Beispiel die von Bitcoin, existiert ein ähnliches Problem unter dem sogenannten 51 % Angriff. Darunter versteht man, dass mehr als die Hälfte der Rechenleistung von einem Angreifer stammt. Somit kann er eine neue Kette erstellen, und sein Kapital auf beiden Ketten ausgeben. Trotzdem wird der Betrag, den er auf beiden Ketten ausgibt, nur einmal von seinem Kapital abgezogen. Der Fachbegriff dazu nennt sich double spending. Bei jeder Blockchain hat ein solcher Angriff immer einen erheblichen Vertrauens- und Wertverlust der dahinterliegenden Kryptowährung zur Folge. Die einzige Verteidigung für solche Angriffe ist die schiere Grösse einer Blockchain. Je mehr Knoten, also Benutzer, desto mehr kostet es Angreifer, 51 % der Rechenpower zu stellen [19].

Diese Art von "neue Knoten erstellen und dann angreifen" ist auch bekannt unter dem Namen Sybil Attack [27]. Falls nun ein Drittel der Knoten im Netzwerk von einem Angreifer stammt, kann dieser verhindern, dass neue Blöcke erstellt werden. Damit kann auch das Entfernen seiner Knoten oder das Hinzufügen legitimer Knoten verhindert werden. Daraus resultiert, dass die gesamte Blockchain nie mehr zu gebrauchen ist. Die Ethereum Proof of Stake Blockchain beschreibt zwei mögliche Verteidigungen gegen diesen Angriff [15], allerdings sind beide nicht auf die Iroha Blockchain anwendbar.

Wenn zwei Drittel der Knoten im Netzwerk vom Angreifer stammen, kann dieser theoretisch jede Aktion in der Blockchain bewilligen. Ein Angriffsszenario mit der Iroha Blockchain könnte wie folgt aussehen:

Der Angreifer entfernt alle Knoten aus dem Netzwerk, ausser seine eigenen und den Knoten n, der sein Ziel ist. Mit diesem Knoten n tätigt er eine Transaktion, das heisst, in die Blockchain wird sinnbildlich niedergeschrieben: Der Angreifer schuldet dem Knoten n etwas. Knoten n validiert das, der Angreifer erhält die Gegenleistung von Knoten n und der neue Block Y wird in die Blockchain aufgenommen, siehe (4).

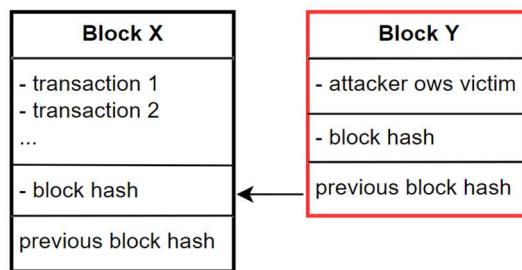


Abbildung 4: Angreifer und Ziel erstellen einen Block

Danach entfernt der Angreifer den Knoten n aus dem Netzwerk und löscht den soeben erstellten Block Y aus seinen Kopien der Blockchain. Dafür erstellt er einen neuen Block Z, mit beliebigem, aber validem Inhalt, siehe Abbildung (5).

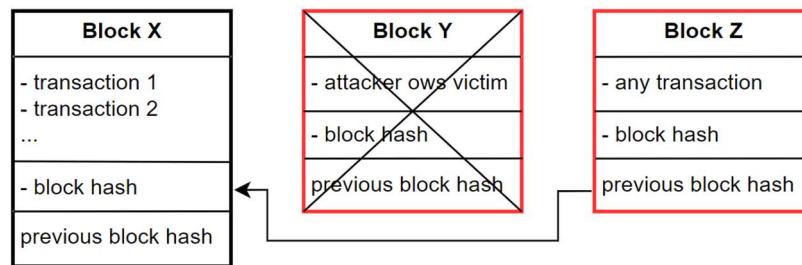


Abbildung 5: Angreifer löscht den Block

Nachdem die anderen Knoten wieder dem Netzwerk beigetreten sind, sei das freiwillig oder vom Angreifer angeordnet, existiert der entfernte Block Y auf keiner Blockchain der Teilnehmer mehr. Der Block Z jedoch wird von allen Teilnehmern als legitimer Nachfolger von Block X angesehen, gemäss Abbildung (6). Nun könnte sogar Knoten n wieder beitreten, jedoch wird niemand seine Blockchain mit Block Y akzeptieren. Somit hat der Angreifer erfolgreich den Eintrag gelöscht und schuldet Knoten n nichts mehr.

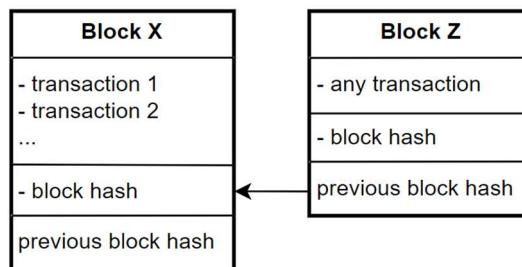


Abbildung 6: Neue, valide Blockchain

Die einzige Verteidigung gegen einen Angriff dieser Art ist, wenn der Angreifer gar nie genügend Knoten kontrollieren kann. Dazu kann definiert werden, dass neue Knoten zusätzliche Anforderungen haben, wie z.B. nur X neue Knoten pro Minute, nur wenige Knoten mit ähnlichen IP-Adressen, usw. Diese Abwehrtechniken sind zum Teil schwer zu implementieren, weil diese auch gegen normale DDoS und

Sybil Angriffe eingesetzt werden, und auch dort überwunden werden können [24]. Eine genauere Analyse davon und detailliertere Massnahmen dagegen sind im Fazit ersichtlich.

2.6.3 Hash Collision

Mit einem Hash können Daten von beliebiger Grösse zusammengefasst werden. Zudem entsteht eine komplett andere Zusammenfassung, wenn die ursprünglichen Daten sich auch nur minimal verändern. Bei einer Blockchain wird, wie in den letzten Kapiteln beschrieben, ein Hash eines Blocks erstellt und diesen in den nächsten Block eingetragen. Somit kann zum einen die Reihenfolge der Blöcke nicht geändert werden, und zum anderen der Inhalt nicht geändert werden, weil der resultierende Hash ein komplett anderer wäre. In der Theorie besteht bei jedem Hash Algorithmus jedoch die Gefahr, dass mehrere unterschiedliche Ausgangsdaten, also unterschiedliche Daten innerhalb des Blocks, zum selben Hash führen können. Dies nennt sich eine Hash Collision.

Als Ausgangslage nimmt sich ein Angreifer einen Block, und versucht nun, die Daten innerhalb des Blockes so zu verändern, um denselben resultierenden Hash zu bekommen. Die Wahrscheinlichkeit, dass er neue Daten findet, die im selben Hash resultieren, kann mit dem Geburtstagsparadoxon erklärt werden [28]. Dieses beschreibt, dass im Durchschnitt bei Gruppen von 23 Personen zwei Personen am gleichen Tag Geburtstag haben. Im Gegensatz zum gleichen Geburtstag ist ein Angreifer nun an einem gleichen Hash interessiert.

Angenommen ein Hash besteht aus 128 Bits: Der Angreifer muss somit nicht durchschnittlich 2^{127} verschiedene Eingangsdaten überprüfen, um denselben Hash zu erhalten, sondern nur etwa 2^{64} . Mit weiteren Optimierungen kann diese Zahl sogar auf 2^{50} verschiedene Eingangsdaten im Durchschnitt reduziert werden, was definitiv nicht mehr als sicher bezeichnet werden kann [29].

Solche Angriffe sind bekannt und existieren schon seit geraumer Zeit. Um einen Angreifer abzuwehren, sind moderne Hashes wie der SHA-2 oder SHA-3 empfohlen. Hash Funktionen wie SHA-1 und vor allem MD5 sind auf jeden Fall zu vermeiden [30] [31].

2.6.4 Replay Attack

Bei einem Replay Angriff wird versucht, eine gültige Aktion ein zweites Mal auszuführen. Dazu muss der Angreifer nur eine gültige Aktion aufzeichnen und diese später nochmals tätigen. Das Aufzeichnen der Aktion ist praktisch immer möglich, auch wenn die Nachricht verschlüsselt ist. Der Empfänger der Nachricht hatte auch die originale Nachricht ebenso verschlüsselt empfangen und entschlüsseln können, somit kann der Angreifer einfach die verschlüsselte Nachricht nochmals schicken.

Als Beispiel mit einer Blockchain: Alice erstellt einen neuen Eintrag für Bob. Danach versendet sie diesen Eintrag an alle Knoten des Iroha Netzwerks. Zeichnet Bob diese Versendung auf, wäre es möglich für ihn, diesen Eintrag einfach nochmals an alle zu versenden. Ohne gewisse Sicherheitsvorkehrungen würden die anderen Knoten im Netzwerk nicht erkennen, dass Alice diesen zweiten Eintrag nicht genehmigt hat.

Als eine der bekanntesten Methoden gegen diese Angriffe ist ein Zeitstempel oder ein Zähler, der mit in den verschlüsselten Teil eines Eintrages einfliest. Dadurch können unabsichtliche Kopien oder eben absichtliche Replay Angriffe erkannt und anschliessend abgelehnt werden [32].

2.6.5 Signature Forging

Um einen neuen Eintrag in einen Block zu bewilligen, muss der Sender diesen Eintrag elektronisch unterschreiben. Das geschieht mit einem Schlüsselpaar, bestehend aus einem privaten und einem öffentlichen Schlüssel. Mit dem privaten Schlüssel, der nur dem Sender bekannt ist, kann ein Eintrag unterschrieben werden. Danach kann mit dem öffentlichen Schlüssel die Signatur "entschlüsselt" werden, und somit die Authentizität des originalen Eintrags bestätigt werden. Wird der Eintrag manipuliert oder mit einem anderen privaten Schlüssel unterschrieben, ergibt die "entschlüsselte" Signatur nicht mehr den originalen Eintrag und ist somit ungültig.

Gelingt es einem Angreifer, den privaten Schlüssel eines Knoten zu lesen oder vom öffentlichen Schlüssel herzuleiten, kann der Angreifer mit diesem Schlüssel neue Einträge erstellen, die nicht mehr von authentischen und legitimen Einträgen des ursprünglichen Benutzers zu unterscheiden sind.

Diese Angriffe können mit einem geeigneten Public-Key Verschlüsselungsalgorithmus verhindert werden. Der Algorithmus muss dabei ordentlich getestet, allgemein akzeptiert, auf dem neusten Stand und mit den empfohlenen Parametern implementiert sein. Beispiele dazu sind RSA, El Gamal und Elliptische Kurven Kryptologie. Für RSA bedeutet das ein 2000, respektive 3000 Bit Modulus, für El Gamal einen Schlüssel mit 250 Bit und eine Gruppenordnung mit 2000, respektive 3000 Bit, und für Elliptische Kurven Kryptologie eine Gruppenordnung von 250 Bit. Diese Empfehlungen gelten für die Jahre 2020-2022, respektive 2023-2026 [33].

2.6.6 Supply Chain Attack

Um DIVA.EXCHANGE anzugreifen, muss nicht unbedingt eine Schwachstelle in DIVA.EXCHANGE selber gefunden werden. Da DIVA.EXCHANGE Iroha und I2P verwendet, kann es unter Umständen reichen, wenn bei diesen zweien eine Schwachstelle existiert. Grundsätzlich wäre auch möglich, dass ein Angreifer den Code von DIVA.EXCHANGE, Iroha oder I2P auf Codeberg oder Docker Hub anpasst, und so neue Schwachstellen erzeugt. Das wäre möglich mithilfe einer Schwachstelle in Codeberg oder Docker Hub selbst, oder wenn sich der Angreifer als legitimen DIVA.EXCHANGE Entwickler ausgibt, oder wenn er das Passwort eines legitimen DIVA.EXCHANGE Entwicklers erratet und so den Code zu seinen Gunsten anpassen kann.

Alternativ kann der Angreifer nach einer Schwachstelle in den verwendeten Frameworks suchen. Dabei ist die Auswirkung je nach Schwachstelle gering bis fatal. Angenommen Iroha speichert IP-Adressen von den teilnehmenden Knoten, und durch eine Schwachstelle sind diese IP-Adressen von aussen einsehbar: Diese Schwachstelle hätte nur geringe Auswirkungen, da I2P trotzdem seine Benutzer anonymisiert und alle Daten authentisch sein müssen. Anderes Beispiel: Wäre es angenommen möglich, um mit einem Admin Account neue Einträge zu generieren, die von den anderen Knoten nicht authentifiziert sein müssen, und der Angreifer findet einen Weg, sich als Admin einzuloggen, ist diese Schwachstelle fatal.

Die Verhinderung solcher Angriffe ist am effektivsten, wenn die Entwickler über solche Supply Chain Attacks Kenntnisse haben. Zudem soll eine übersichtliche und komplettete Liste der verwendeten externen Software erstellt werden. Damit können direkt Massnahmen eingeleitet werden, wenn bei einer verwendeten Software oder einer verwendeten Library eine Schwachstelle entdeckt wird, sei das ein Update oder das vorübergehende Abschalten von kritischen Services [34].

3 Methodik

Das Vorgehen zum Feststellen und Klassifizieren von Schwachstellen wird in diesem Kapitel beschrieben. Zuerst werden dazu die möglichen Schwachstellen ausgemacht, danach wird aufgezeigt, wie der Teststand funktioniert und schlussendlich wird beschrieben, wie die Ergebnisse entstehen und zu interpretieren sind.

3.1 Sicherheitskritische Aspekte aufdecken

In diesem Unterkapitel wird die Herangehensweise beschrieben, wie DIVA.EXCHANGE auf hoher Ebene funktioniert, wie mögliche Schwachstellen ausgemacht werden, und wie getestet wird, dass es tatsächlich Schwachstellen sind.

Dazu werden als erstes Abuse-Cases dokumentiert. Diese unterscheiden sich von normalen Use-Cases, indem der Angreifer miteinbezogen wird und erste potenzielle Schwachstellen der Anwendung offen gelegen werden. Nachfolgend wird mit einem Data-Flow-Diagramm der Aufbau von DIVA.EXCHANGE genauer gezeigt. Mit der gewonnenen Übersicht aus dem Data-Flow-Diagramm wird ein Threat-Modelling durchgeführt. Dabei werden erneut potenzielle Schwachstellen, also Angriffspunkte, gesucht und dokumentiert.

Die ausgemachten Schwachstellen werden in den Abschnitten Abuse-Case und Threat-Modelling genauer analysiert und kategorisiert. Diese Kategorien sind wie folgt: Eine potenzielle Schwachstelle kann als "wird nicht untersucht" bezeichnet werden. Die Definition davon befindet sich im nächsten Abschnitt. Auch möglich ist, dass direkt gezeigt werden kann, dass die potenzielle Schwachstelle keine relevanten Auswirkungen hat, oder gar keine wirkliche Schwachstelle ist.

Die letzte Kategorie ist, dass die potenzielle Schwachstelle genauer untersucht werden muss. Dazu kann sie mit **S<Nr.>, T<Nr.> oder R<Nr.>** genauer gekennzeichnet werden. S steht für "theoretisch oder statisch analysieren", also den Code unter die Lupe nehmen und so entscheiden, ob es eine Schwachstelle ist, und wie gravierend ihre Auswirkungen sind. T steht für Test, also die potenzielle Schwachstelle mit einem Test untersuchen und damit darüber urteilen. Die Resultate der statischen Analyse und der Tests sind in Kapitel 4 ersichtlich. R steht für "Risiko minimieren". Das bedeutet, dass die potenzielle Schwachstelle ohne weiteres Analysieren als Schwachstelle aufgenommen und dokumentiert wird. Dies aus dem Grund, weil das detaillierte Untersuchen zwar zu aufwändig oder gar unmöglich ist, aber die ungefähren Auswirkungen und Gegenmassnahmen bekannt sind und das DIVA.EXCHANGE Entwicklerteam darüber informiert sein muss.

3.1.1 "wird nicht untersucht" - Definition

Ein eventueller sicherheitskritischer Aspekt ist mit "wird nicht untersucht" bezeichnet, wenn dieser ein oder mehrere der folgenden Punkte erfüllt:

- Der Aspekt betrifft eine mögliche Schwachstelle in oder mithilfe von I2P.
- Der Aspekt betrifft eine mögliche Schwachstelle in oder mithilfe von Docker Hub oder Codeberg.
- Der Aspekt kann aus zeitlichen Gründen nicht genauer untersucht werden, weil andere Aspekte einen grösseren Wert zur genaueren Untersuchung bieten, zum Beispiel die Diva-API im Vergleich zur Iroha-API.

3.1.2 Abuse-Cases

Das untenstehende Abuse-Case-Diagramm (7) zeigt auf, welche Angriffspunkte grundsätzlich in Frage kommen. Der Angreifer und die möglichen Angriffe sind in der oberen Hälfte der Abbildung ersichtlich. Die legitimen Benutzer sind mit ihren Use-Cases in der unteren Hälfte zu sehen. Diese Use-Cases und die dazugehörigen Angriffe werden auf der nächsten Seite beschrieben.

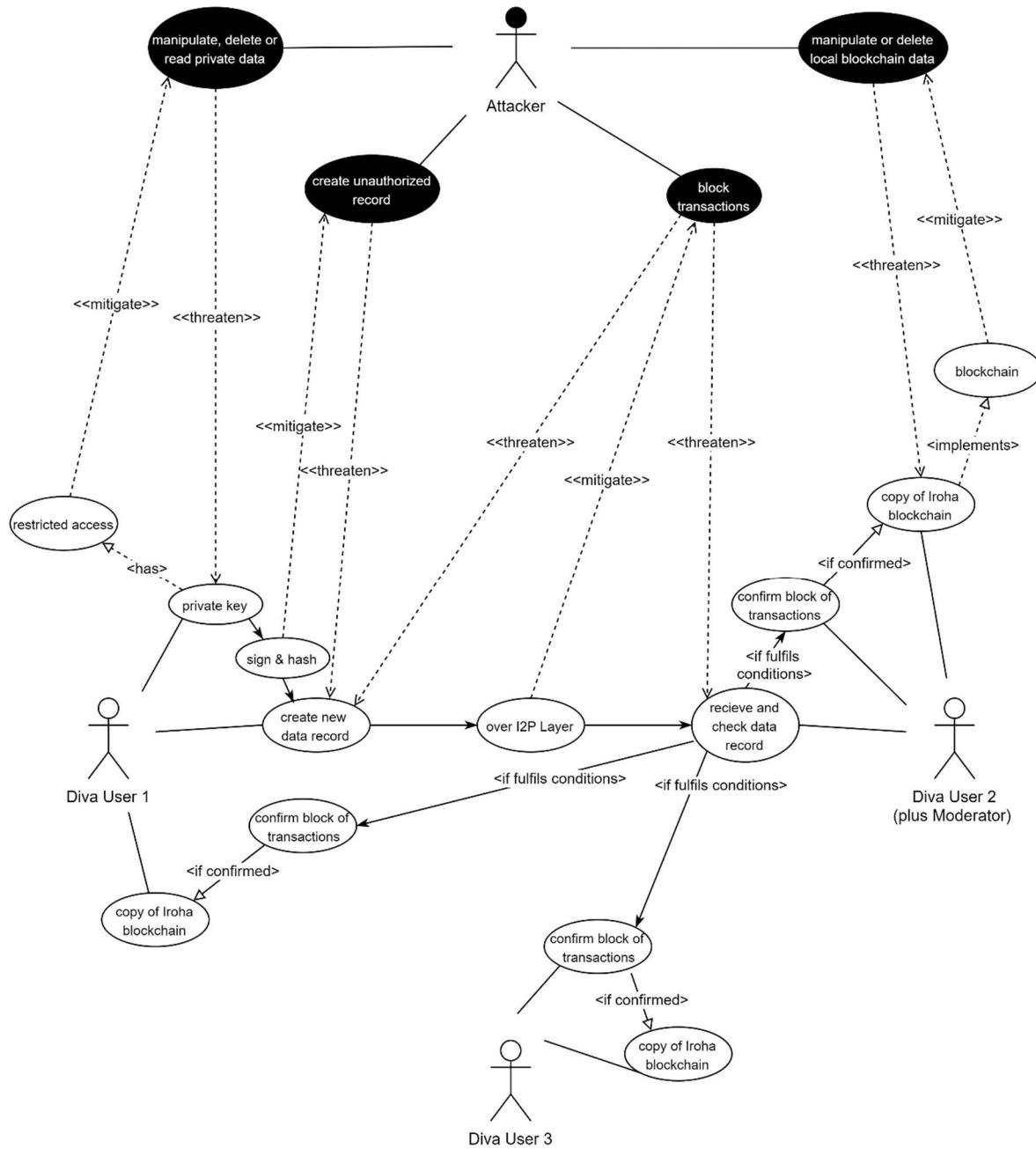


Abbildung 7: Abuse-Case Diagramm

3.1.2.1 Neue Daten zur Blockchain hinzufügen

Um neue Daten zur Blockchain hinzufügen, muss ein DIVA.EXCHANGE Knoten mit den anderen Knoten kommunizieren. Würde der Angreifer diese Kommunikation blocken, wäre Diva nicht mehr nutzbar. Die Kommunikation erfolgt allerdings über das I2P Netzwerk. Einerseits anonymisiert I2P die Knoten der Benutzer und macht es für einen Angreifer viel schwieriger, die Kommunikation zu blockieren [25] [35]. Andererseits ist I2P als "wird nicht untersucht" klassifiziert für diese Arbeit.

- ➔ Dieser Aspekt wird nicht weiter untersucht.

3.1.2.2 Neue Einträge in einen Block authentifizieren

Bevor die DIVA.EXCHANGE Knoten über einen neuen Block abstimmen können, müssen die Benutzer der Knoten erst einmal die Daten erstellen, die in den neuen Block eingefügt werden. Diese Daten müssen vom Ersteller signiert werden, andernfalls kann nicht nachgewiesen werden, ob ein Eintrag wirklich von einem gewissen Benutzer, also Knoten, erstellt worden ist, Stichwort "Signature Forging". DIVA.EXCHANGE muss eine asymmetrische Verschlüsselung benutzen, die generell als sichere Verschlüsselung akzeptiert ist. Zudem muss diese in sicherer Art und Weise implementiert sein, Stichwort "API Abuse".

- ➔ **S1** Es muss untersucht werden, welche Verschlüsselung verwendet wird und wie diese implementiert ist.
- ➔ **R1** Eigene, veraltete aber auch weniger verbreitete Verschlüsselungsalgorithmen dürfen nicht verwendet werden.

Für die oben genannte asymmetrische Verschlüsselung benötigt jeder Knoten einen privaten und einen öffentlichen Schlüssel. Der private Schlüssel darf nicht für anderen Knoten zugänglich sein. An dernfalls kann erneut nicht garantiert werden, dass die Daten in der Blockchain authentisch sind. Als Beispiel: Wenn Bob den privaten Schlüssel von Alice kennt, könnte er Daten erstellen, aber behaupten Alice sei die Erstellerin. Die Daten sehen nun für alle so aus, als kämen sie wirklich von Alice. Diese Schwachstelle könnte erneut wegen potenziellem API-Abuse existieren.

- ➔ **S2** Es muss untersucht werden, ob der private Schlüssel nicht für andere Knoten zugänglich ist.

3.1.2.3 Blockchain Status aktualisieren

Nachdem die DIVA.EXCHANGE Knoten einen Konsens bei einem neuen Block erreichen, wird dieser in die lokalen Kopien der Blockchain aufgenommen. Das heisst, wird erfolgreich ein neuer Block erstellt und angenommen, aktualisiert jeder Knoten seine Blockchain. DIVA.EXCHANGE wird in einem Docker Container ausgeführt, und man braucht root-Rechte, also Administrationsrechte, um die Blockchain manuell anzupassen. Allerdings kann durch einen eventuellen Bug oder eine Schwachstelle in der API ermöglicht werden, die lokale Blockchain via API zu manipulieren.

- ➔ **S3** Es muss untersucht werden, ob die lokale Blockchain nur durch akzeptierte Blöcke angepasst werden kann.

3.1.3 Data-Flow-Diagramm

Das untenstehende Diagramm (8) zeigt den Aufbau von DIVA.EXCHANGE. Ersichtlich ist, wie die einzelnen Komponenten von DIVA.EXCHANGE miteinander kommunizieren. Aus Gründen der Übersichtlichkeit werden nur zwei komplette DIVA.EXCHANGE Prozesse abgebildet, in Realität kommunizieren viel mehr DIVA.EXCHANGE Knoten über das I2P Netzwerk miteinander.

Im oberen Teil des Diagramms ist ersichtlich, dass die Benutzer der Knoten als externe Entität über das Diva Userinterface auf die Diva-API zugreifen. Die von der roten Linie umgebenen Prozesse sind gesicherte Zonen, und somit mehr abgeschirmt und geschützt als der Rest der Applikation. Die Diva-API kann auf den Speicher des Knoten zugreifen, sowie auf den kompletten Iroha Prozess. Dieser beinhaltet unter anderem die Iroha-API, und verwaltet die lokale Blockchain. Mithilfe von GRPC und dem I2P Netzwerk können die Iroha Knoten untereinander komplett anonymisiert kommunizieren [35].

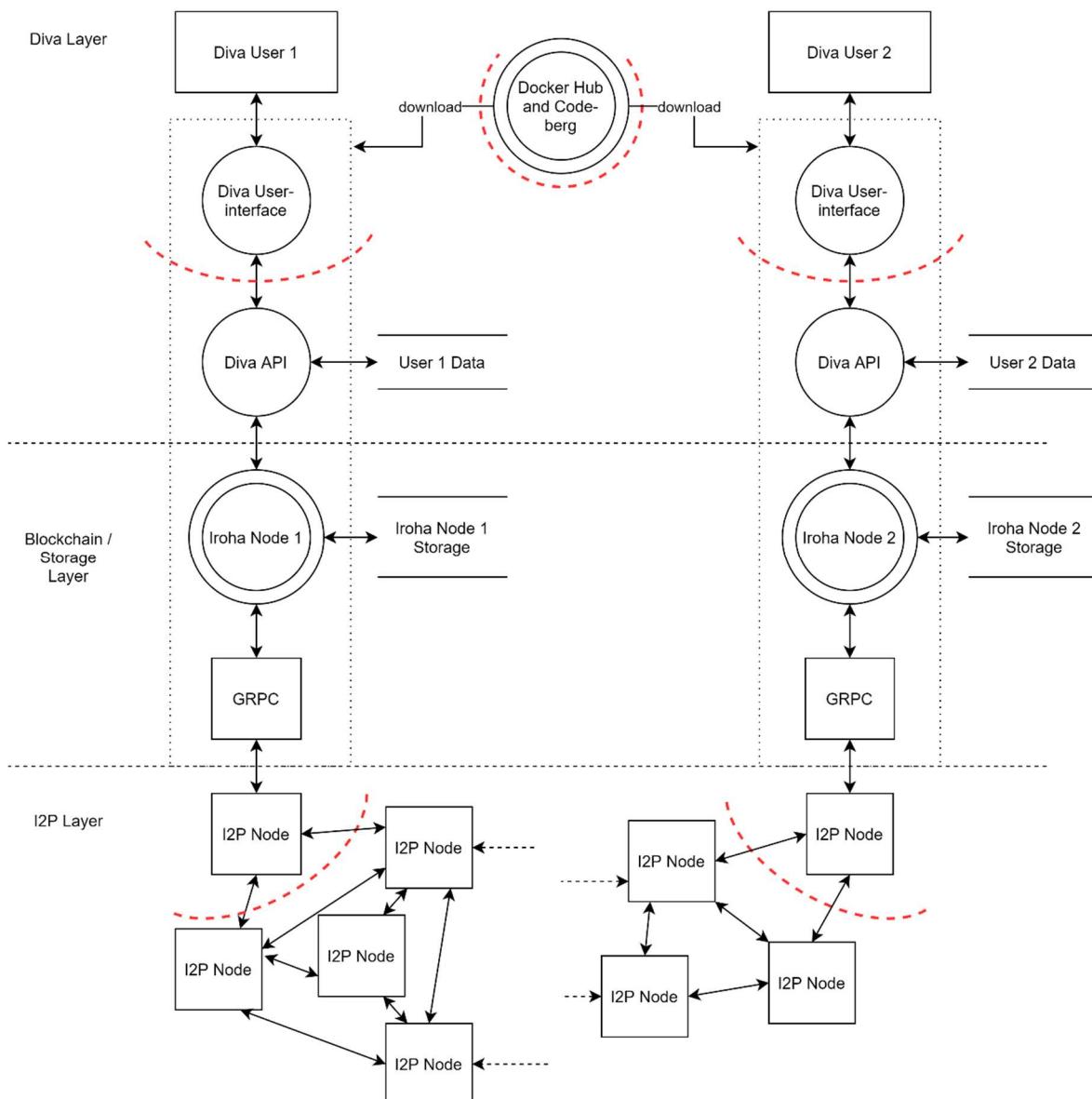


Abbildung 8: Data-Flow-Diagramm

3.1.4 Threat-Modelling

Folgend werden spezielle Ausschnitte des Data-Flow-Diagramm genauer untersucht. Diese Ausschnitte enthalten Funktionen und Abläufe, die essenziell sind um ein korrektes und sicheres Verhalten der DIVA.EXCHANGE Plattform sowie der Iroha Blockchain zu garantieren. Auch hier wird bei jedem Ausschnitt analysiert, ob eine potenzielle Schwachstelle vorhanden ist, und wie diese weiter untersucht werden soll.

3.1.4.1 Daten manipulieren oder Kommunikation blockieren zu einzelnen Benutzer

In der untenstehenden Abbildung (9) ist ersichtlich, wie ein Angreifer alle I2P Nachbarknoten eines legitimen Benutzers kontrolliert. Zum Ersten ist das unrealistisch, weil I2P anonymisiert ist und somit der Angreifer nicht gezielt die Nachbarn eines bestimmten Knoten kontrollieren kann. Zum Zweiten kann der Angreifer die Daten nicht unbemerkt verändern, da alle Daten von den Erstellern authentifiziert sind. Er könnte nur die Kommunikation unterbrechen, sollte ihm gelingen, alle Nachbarknoten zu kontrollieren. Zum Dritten ist I2P als "wird nicht untersucht" klassifiziert.

→ Dieser Aspekt wird nicht weiter untersucht.

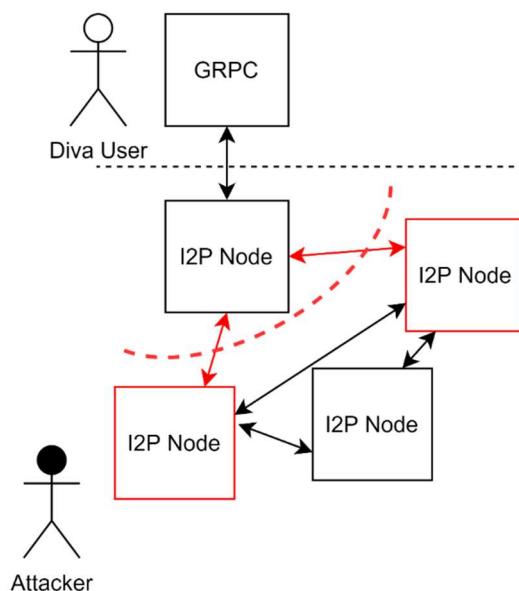


Abbildung 9: Angriff via I2P

3.1.4.2 Codebase von DIVA.EXCHANGE und Iroha manipulieren

Unten abgebildet (10) ist, wie ein Angreifer den Code von DIVA.EXCHANGE oder Iroha zu seinem Nutzen verändert, Stichwort Supply Chain Attack. Das Ausnutzen von existierenden Schwachstellen wird bei den Abbildungen 11, 12 und 13 untersucht. Schwachstellen in Codeberg und Docker Hub sind als "wird nicht untersucht" klassifiziert. Allerdings kann ein Angreifer sich auch als DIVA.EXCHANGE Entwickler ausgeben, um absichtlich neue Schwachstellen in den Code einzubauen. Alternativ kann er auch mit den Login Daten eines legitimen DIVA.EXCHANGE Entwicklers den Code von DIVA.EXCHANGE anpassen.

- ➔ **R2** Es müssen sichere Passwörter [36] von allen DIVA.EXCHANGE Entwickler verwendet werden und der Code von Entwickler, vor allem von neuen Entwicklern, muss regelmässig überprüft werden.

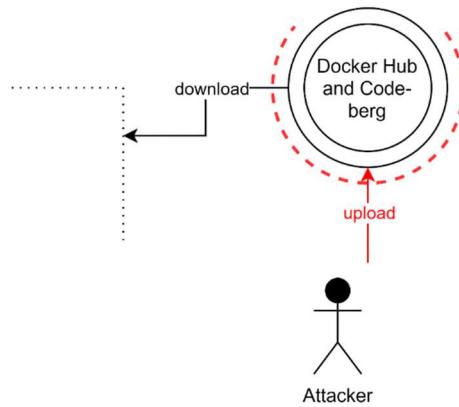


Abbildung 10: Supply Chain Attack via Docker Hub oder Codeberg

3.1.4.3 Lokale Blockchain eines Benutzers manipulieren

Nachstehend ist ersichtlich (11), wie ein Angreifer die lokalen Blockchain Daten eines anderen Knoten manipuliert. Dabei wäre möglich, dass er ein oder mehrere Blöcke der Blockchain löscht oder anpasst, um seine belastenden Transaktionen zu verdecken.

- ➔ Dieses Szenario ist in den Abuse-Cases dokumentiert und hat die Kennzeichnung **S3**.

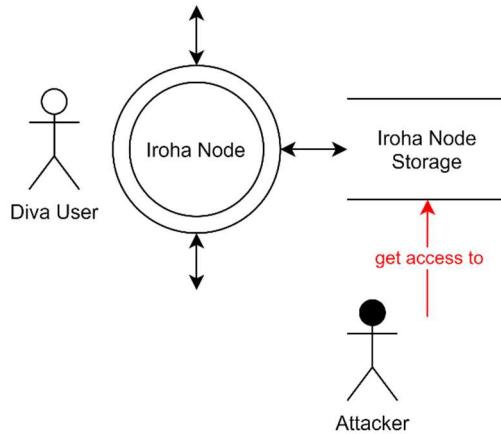


Abbildung 11: Angriff auf lokale Blockchain

3.1.4.4 Lokale Daten eines Benutzers manipulieren

Wie im vorherigen Angriffsszenario versucht der Angreifer auf private Daten eines Knoten zuzugreifen (12). Sollte ihm das gelingen, kann er Daten zu neuen Blöcken hinzufügen, die nicht zu unterscheiden sind von denen, die der legitime Benutzer des Knoten veröffentlicht. Andererseits könnte er den privaten Schlüssel des Knoten löschen, was es dem Benutzer verunmöglicht, seinen Knoten weiterhin zu benutzen, ausser der angegriffene Benutzer hat ein Backup von seinem privaten Schlüssel, und die nötigen Kenntnisse, um seinen Knoten wieder zu reparieren.

→ Dieses Szenario ist in den Abuse-Cases dokumentiert und hat die Kennzeichnung **S2**.

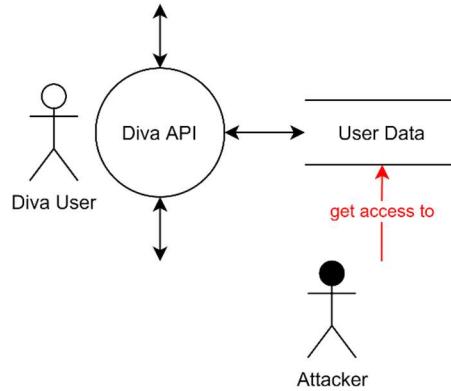


Abbildung 12: Angriff auf lokale Daten

3.1.4.5 Anderen Benutzer imitieren

Gelingt es einem Angreifer, den PC eines DIVA.EXCHANGE Benutzers unter Kontrolle zu bringen, ist es ihm theoretisch möglich, sich als diesen Benutzer, also diesen Knoten auszugeben (13). Dazu kann er Schwachstellen in anderen Programmen ausnutzen, oder physisch an den PC gelangen. Allerdings wird DIVA.EXCHANGE als root ausgeführt, was bedeutet, dass man sich als root (Administrator) authentifizieren muss, bevor man darauf zugreifen kann. Wenn sich ein Angreifer aber als root authentifizieren kann, liegt die Schwachstelle nicht bei DIVA.EXCHANGE. Zudem könnte dies nur gelöst werden, wenn DIVA.EXCHANGE eine zentrale Autorität hätte, bei der man sich authentifizieren muss, bevor man neue Daten auf die Blockchain geben kann. Das widerspricht aber direkt dem Ziel von DIVA.EXCHANGE.

→ Dieser spezifische Angriff wird nicht weiter untersucht. Die relevanten Schwachstellen davon sind schon in **S2** enthalten.

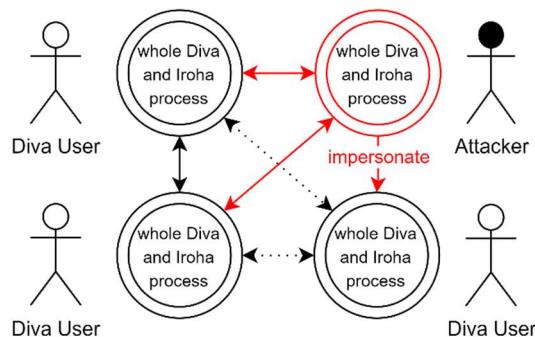


Abbildung 13: Anderen Benutzer imitieren

3.1.4.6 Neue Einträge in die Blockchain verhindern

In der untenstehenden Abbildung (14) wird ein Angriff gezeigt, der die Benutzung von DIVA.EXCHANGE für legitime Benutzer verunmöglicht. Eine Schwäche des verwendeten YAC Algorithmus [17] ist, dass keine neuen Blöcke angenommen werden können, wenn die Angreifer einen Dritteln oder mehr der Knoten kontrollieren, Stichwort Liveness Denial. Dies, weil eine gewisse Mehrheit der Knoten die neuen Blöcke annehmen muss, um die von YAC definierte byzantinische Fehlertoleranz zu erreichen. Wie im Kapitel "2.6 Angriffe" erörtert, gibt es viele Herausforderungen, um die Knoten von solchem Angreifer wieder aus dem Netzwerk auszuschliessen. Dazu kommt, dass erstens DIVA.EXCHANGE gratis ist und jeder sich die Software herunterladen kann, zweitens nur ein Drittel aller Knoten von einem Angreifer stammen müssen, gemäss den YAC Spezifikationen jedenfalls [17], und drittens die Knoten der Angreifer nur sehr wenig Rechenleistung benötigen, um das DIVA.EXCHANGE Programm laufen zu lassen, siehe "Docker Ressourcenverbrauch" im Anhang. Das alles führt dazu, dass ein einzelner Angreifer mit wenigen Ressourcen viele Knoten kontrollieren kann, auch bekannt unter Sybil Attack.

- ➔ **T1** Es muss gezeigt werden, wie viele Knoten ein Angreifer für ein bestimmtes Netzwerk kontrollieren muss, bevor keine neuen Einträge in die Blockchain aufgenommen werden. Zudem soll grob aufgezeigt werden, wie viele Ressourcen ein Angreifer benötigt, um diesen Anteil an Knoten zu besitzen.

Falls sich neue Knoten zuerst bei einer zentralen Autorität authentifizieren müssen, kann diese kontrollieren, wer dem Netzwerk beitritt, oder sogar verdächtige Knoten wieder aus dem Netzwerk ausschliessen ohne die Zustimmung anderer Knoten. Somit wäre dieser Angriff nicht möglich. Allerdings widerspricht eine zentrale Autorität direkt dem Konzept einer Blockchain, und auch dem Ziel von DIVA.EXCHANGE, ein vollständig verteiltes Handelsnetzwerk zu sein [1].

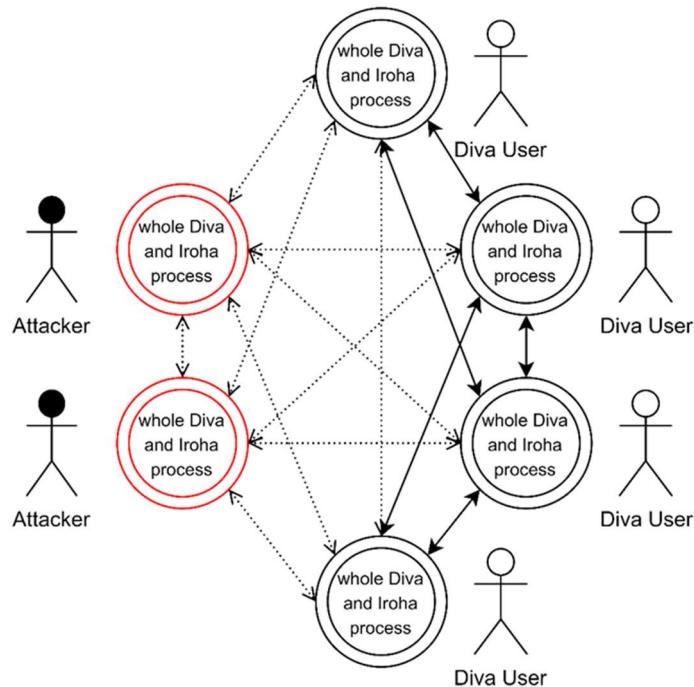


Abbildung 14: Neue Einträge blockieren

3.1.4.7 Blockchain manipulieren mithilfe Netzwerkangriff

Auf der folgenden Abbildung (15) ist zu sehen, wie die Angreifer die Knoten von legitimen Benutzern vom Netzwerk ausschliessen, um eine genügend grosse Mehrheit erzielen. Damit können sie beliebige neue Blöcke in die Blockchain aufnehmen. Wie im Kapitel "2.6 Angriffe" im Abschnitt DDoS aufgezeigt, ist es keine einfache Aufgabe, ein Optimum zwischen

- a) Knoten zeitnah entfernen, die nicht mehr antworten, um die Abstimmung nicht zu blockieren, und
- b) Knoten auch bei Inaktivität eine Zeit im Netzwerk lassen, um grosse Schwankungen an aktiven Knoten zu vermeiden,

zu finden.

Alternativ kann auch mit dem Ansatz gearbeitet werden, dass die Knoten nie durch Inaktivität ausgeschlossen werden. Somit wäre dieser Angriff in dieser Form nicht möglich, andererseits würde der vorherige Angriff "Neue Einträge in die Blockchain verhindern" um einiges einfacher werden, weil nun die Knoten nur einmal registriert werden müssen, und dann einfach ausgeschalten werden könnten.

→ **T2** Es muss gezeigt werden, ob und wie möglich ist, die Knoten vom Netzwerk zu entfernen, und wie lange dies dauern würde, respektive wie viele oder wer einer solchen Entfernung zustimmen muss.

Wie beim letzten Angriff gilt: Mit einer zentralen Autorität kann dieser Angriff verhindert werden. Erneut widerspricht eine solche Autorität direkt dem Konzept einer Blockchain, und dem Ziel von DIVA.EXCHANGE [1].

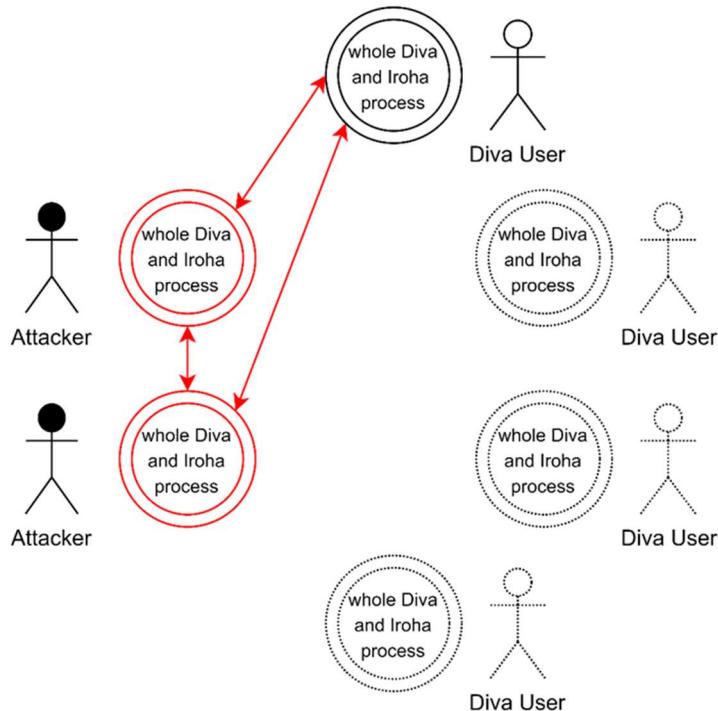


Abbildung 15: Legitime Knoten ausschliessen

3.1.4.8 Blockchain manipulieren mithilfe gewisser Mehrheit

Ähnlich wie beim "Neue Einträge in die Blockchain verhindern" Angriff, versuchen hier (16) die Angreifer eine gewisse Anzahl von Knoten zu kontrollieren. Jedoch nicht, um das Netzwerk lahmzulegen, sondern um selbst Einträge in die Blockchain zu generieren.

- ➔ **T3** Es muss gezeigt werden, wie viele Knoten ein Angreifer für ein bestimmtes Netzwerk kontrollieren muss, damit er beliebige Einträge in die Blockchain aufnehmen kann. Auch hier soll zudem grob aufgezeigt werden, wie viele Ressourcen ein Angreifer benötigt, um diesen Anteil an Knoten zu besitzen.

Auch hier gilt wie bei den beiden letzten Angriffen, dass eine zentrale Autorität diesen Angriff verhindern könnte, aber diese zentrale Autorität dem Konzept einer Blockchain und dem Ziel von DIVA.EXCHANGE direkt widerspricht [1].

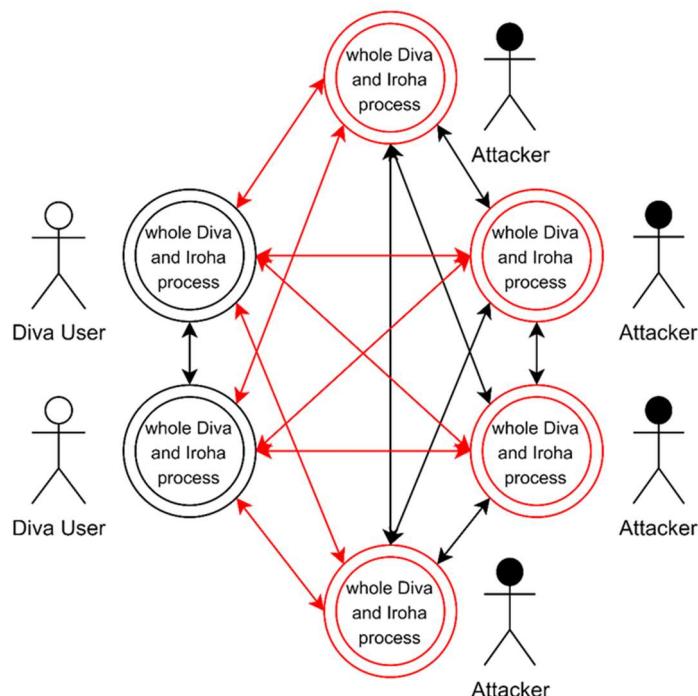


Abbildung 16: Angriff mithilfe gewisser Mehrheit

3.2 Teststand – Aufbau bis Auswertung

Für den Teststand wird das diva-dockerized Testnetz als Vorbild verwendet [37]. Folgend werden der Aufbau und die Funktionsweise des angepassten Testnetzwerks gezeigt (17) und erklärt.

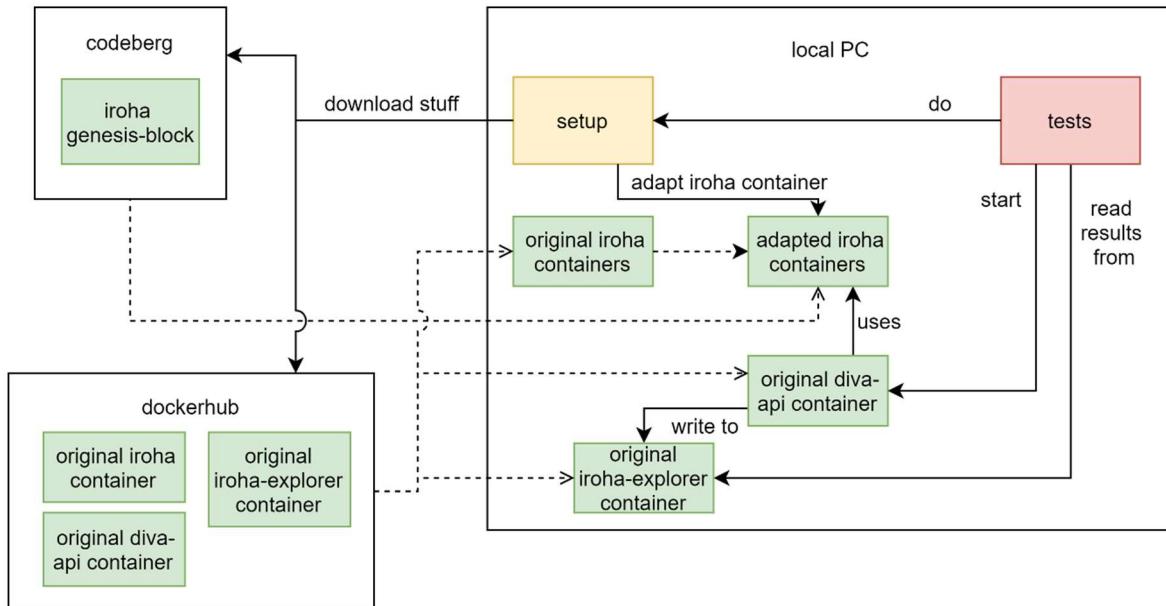


Abbildung 17: Teststand, Aufbau und Ablauf

3.2.1 Erstellen und Anpassen der Knoten

Als erstes werden die Diva und Iroha Container heruntergeladen, sowie der Iroha Genesis-Block, also der erste Block für die kommende Blockchain. Nachdem alle Dateien auf dem lokalen Rechner sind, wird die Konfiguration für das Testnetz geschrieben. Dabei kann die Anzahl der Knoten, die beim Testnetzwerk dabei sind, festgelegt werden.

Ist die Option "Benchmark" gesetzt, endet nun die Konfiguration, und das Testnetzwerk startet. Die Diva-API und der Iroha-Explorer sind nicht erstellt worden und starten somit auch nicht. Mit der Benchmark soll untersucht werden, wie viele Ressourcen für eine gewisse Anzahl Knoten gebraucht werden. Für einen Angreifer reicht es somit theoretisch, nur die Iroha Container starten zu können.

Ist die Option "Benchmark" nicht gesetzt, müssen die oben erstellten Iroha Container noch leicht angepasst werden. Dabei wird bei allen Iroha Containern die öffentlichen Schlüssel der neuen Knoten hinzugefügt. Also die Schlüssel der Knoten, die nicht zu den ursprünglichen 7 Knoten gehören. Zudem erhält jeder der neuen Iroha Containern noch seinen privaten Schlüssel. Auch der angepasste Genesis-Block wird in alle Iroha Containern kopiert, sodass von Beginn an alle neuen Knoten im Netzwerk dabei sind. Theoretisch ist es auch möglich, neue Knoten erst nach dem Aufstarten des Netzwerks einzufügen. Allerdings dauert das Einrichten für die Tests somit mindestens drei Minuten länger, weil bei jedem Aufstarten für jeden Test noch gewartet werden muss, bis die Knoten hinzugefügt werden. Zudem hat diese Herangehensweise auch nach einigen Versuchen noch nicht funktioniert. Deshalb ist die erste Variante implementiert und in den folgenden Tests sind die neuen Knoten somit schon von Beginn an im Netzwerk.

Sind nun alle Iroha Container fertig eingerichtet, können diese gestartet werden. Für eine ausführlichere Beschreibung des Codes und der genauen Vorgehensweise ist die Code Dokumentation der Tests im Anhang oder das Codeberg Repo zu konsultieren.

3.2.2 Starten des Netzwerks

Sobald alle Iroha Container fertig erstellt und angepasst sind, werden diese sowie die Diva-API und Iroha-Explorer gestartet. Dabei dauert es bei 15 Knoten ca. 10 Sekunden, um die Container aufzustarten. Danach müssen nochmals etwa 15 bis 25 Sekunden gewartet werden, bis alle Knoten ihre initialen Abläufe erledigt haben, also zum Beispiel das Starten von Iroha, das Laden der Konfigurationsdaten und so weiter. Innerhalb dieser Zeit wird periodisch geprüft, ob die Diva-API und der Explorer schon online sind. Sobald diese online sind, wird das Netzwerk als voll funktionsfähig betrachtet, und die Tests können starten.

3.2.3 Ablauf der Tests

Bei den Tests 1 und 3 wird mit dem Ping der Diva-API geprüft, ob das Netzwerk noch funktionsfähig ist. Dieser Ping wird ungefähr jede Minute erzeugt, und wenn genügend Iroha Knoten diesen validieren, wird dieser in die Blockchain aufgenommen. Validieren nun zu wenige Knoten diesen Ping, wird dieser nicht in die Blockchain aufgenommen.

Beim zweiten Test wird auf eine Antwort der Diva-API gewartet. Kann der Knoten entfernt werden, erhält man nach der Entfernung sofort eine Bestätigung. Kann er nicht entfernt werden, antwortet die Diva-API gar nicht.

3.2.4 Auswertung der Tests

Das Testprogramm kann, wie in der Abbildung auf der vorherigen Seite gezeigt, den Stand der Blockchain direkt beim Iroha-Explorer abfragen. Beim Test 1 und 3 wartet das Programm, bis ein Ping in die Blockchain aufgenommen wird, oder bis eine gewisse Zeit abgelaufen ist, auch Timeout genannt. Ist die Zeit abgelaufen, wird mit keinem neuen Ping gerechnet und dies so notiert. Das Eintreffen, oder eben nicht-Eintreffen dieser Pings wird für jede Testrunde festgehalten und am Ende des Tests angezeigt und in einer Datei gespeichert. Eine Testrunde bedeutet zum Beispiel: Es sind 11 Knoten eingeschaltet bei einer Netzwerkgrösse von 15 Knoten insgesamt, und es auf den nächsten Ping oder das Timeout gewartet. Eine weitere Runde wäre dann zum Beispiel, bei 10 eingeschalteten Knoten auf den folgenden Ping oder das Timeout zu warten.

Beim zweiten Test wird anstatt auf einen Ping, auf eine Bestätigung der Diva-API gewartet. Wie bei den anderen Tests wird dann die Bestätigung oder nicht-Bestätigung für jede Testrunde festgehalten, und schlussendlich die Resultate angezeigt und in einer Datei gespeichert.

3.3 Tests – Beschreibung

Im folgenden Abschnitt werden die Tests genauer beschrieben, die im vorherigen Teil als sicherheitskritisch definiert sind. Allerdings werden hier nur die **T1, T2, und T3** beschrieben, die anderen Aspekte, also **S und R**, müssen nicht praktisch analysiert werden. Zudem werden die Auswirkungen und Verhinderungsmassnahmen hier nicht genannt, diese folgen in Kapitel 5.

3.3.1 Test T1 – Neue Einträge in die Blockchain verhindern

Es wird angenommen, dass bei einer genügend grossen Anzahl von Knoten, die nicht an der Abstimmung teilnehmen, keine neuen Blöcke in die Blockchain niedergeschrieben werden, Stichwort DoS Attack und Liveness Denial. Dabei spielt es konzeptionell keine Rolle, ob die Knoten komplett online sind, aber absichtlich nicht abstimmen, oder die Knoten ausgeschalten sind.

Im Test wird deshalb simuliert, dass nacheinander Knoten ausgeschaltet werden, und notiert, bei wie vielen Knoten keine neuen Einträge registriert werden. In der untenstehenden Abbildung (18) bilden die roten Knoten die Angreifer ab, und die anderen die der legitimen Benutzer. Dabei stimmen die Knoten der Angreifer nicht mehr ab, und die Knoten der legitimen Benutzer versuchen den Block trotzdem in die Blockchain aufzunehmen.

Den Block, über dessen Aufnahme abgestimmt wird, ist ein einfacher Ping. Dieser sogenannte Ping wird jede Minute von der Diva-API ausgelöst, und hat dieselben Anforderungen wie jeder andere Eintrag, um in die Blockchain aufgenommen werden, siehe "Ping call stack" im Anhang. Beim allerersten Ping sind noch alle Knoten eingeschaltet, um die Bereitschaft des Netzwerks zu überprüfen.

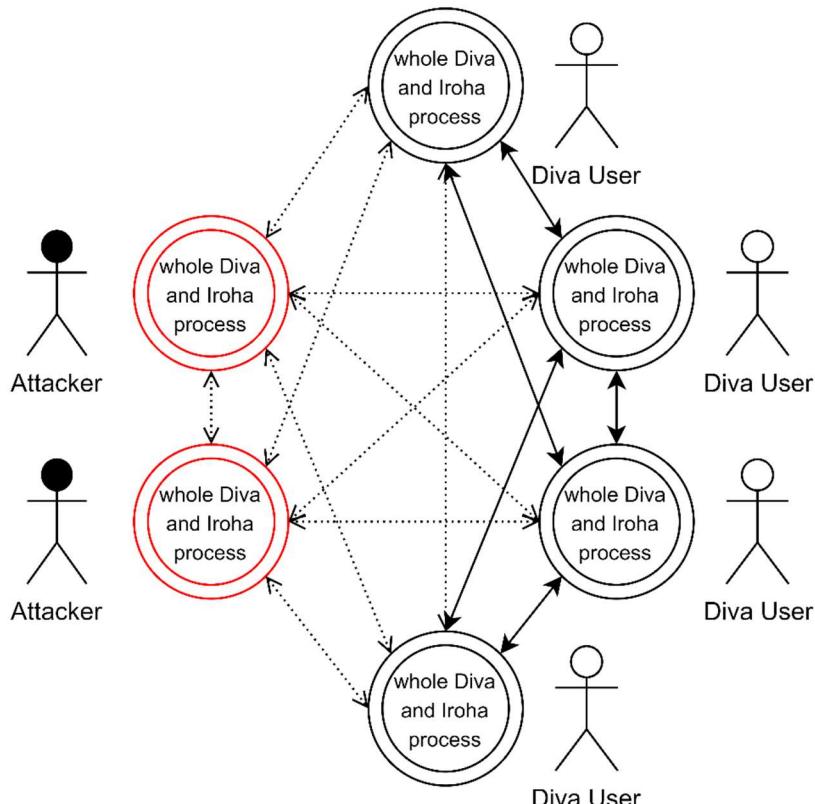


Abbildung 18: Neue Einträge blockieren

Mit diesem Ansatz kann geprüft werden, wie viele Knoten, oder besser gesagt, welcher Anteil der Knoten bei einer Abstimmung antworten muss, damit ein neuer Block noch aufgenommen wird. Zudem kann bei jeder Testrunde (also zuerst nur ein Knoten ausgeschaltet, dann zwei Knoten ausgeschaltet, usw.) notiert werden, wie viele Knoten den Ping validieren. Als Beispiel, mit einer Netzwerkgrösse von 15 müssen gemäss YAC Spezifikation [17] nur 11 Knoten einen neuen Block validieren, auch wenn 12, 13, 14 oder 15 Knoten an der Abstimmung teilnehmen könnten.

3.3.1.1 Erwartetes Resultat

Wie im Bericht von YAC [17] beschrieben, basiert ihr System auf der byzantinischen Fehlertoleranz, also einer $3f+1$ Fehlertoleranz. Eine $3f+1$ Fehlertoleranz bedeutet, dass mehr als zwei Drittel einem neuen Block zustimmen müssen. Bei einem Netzwerk mit 15 Knoten sind das also mindestens 11 Knoten. Somit sollten Pings in die Blockchain eingetragen werden, solange die Anzahl der ausgeschalteten Knoten zwischen 0 und 4 ist. Sind 5 oder mehr Knoten ausgeschalten, sollten keine weiteren Pings auf der Blockchain erscheinen. Wäre Iroha nach $2f+1$ implementiert, also nur eine einfache Mehrheit, müssen bei 15 Knoten nur 8 antworten. Das heisst es könnten bei $2f+1$ zwischen 0 bis 7 Knoten ausgeschalten sein, und der Ping würde immer noch eingetragen werden.

3.3.2 Test T2 – Blockchain manipulieren mithilfe Netzwerkangriff

Die ursprüngliche Idee für diesen Test war, dass wie beim vorherigen Test eine beliebige Anzahl Knoten zu einem bestimmten Zeitpunkt ausgeschalten werden. Hier warten die Angreifer allerdings nach dem Ausschalten nun, bis die ausgeschalteten Knoten nicht mehr im Netzwerk registriert sind, und versuchen dann, neue Einträge in die Blockchain aufzunehmen. Die restlichen legitimen Knoten, die noch eingeschalten sind, versuchen diese Einträge abzulehnen, siehe (19). Es wird angenommen, dass das Ausschalten äquivalent ist zu einem längeren Netzwerkausfall.

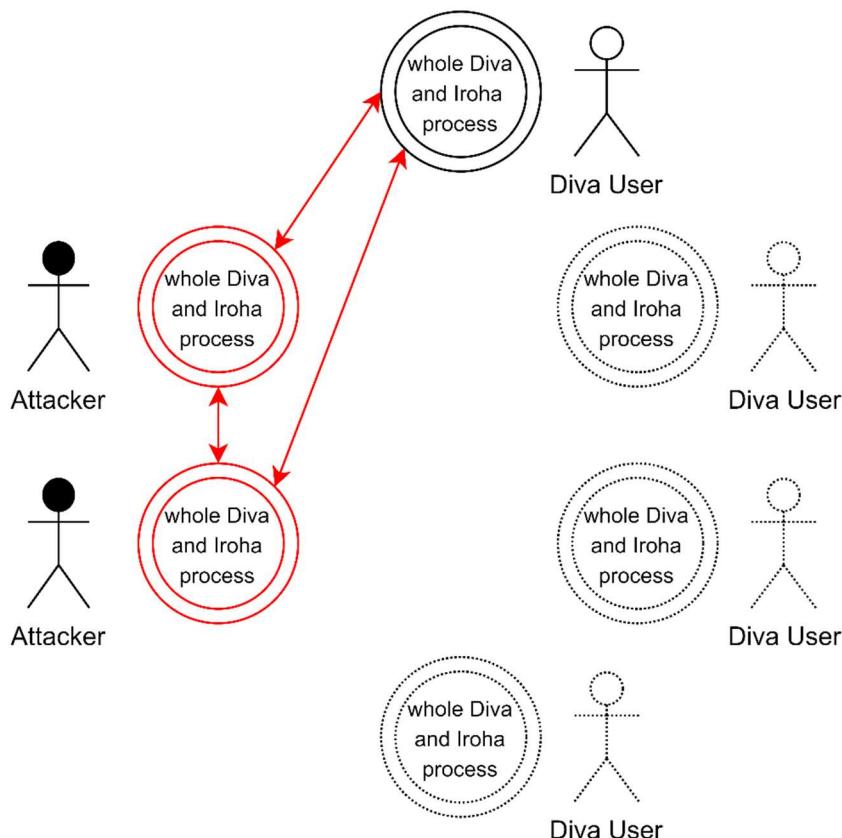


Abbildung 19: Legitime Knoten ausschliessen

Allerdings stellt sich während dem Entwickeln der Tests heraus, dass die Knoten, sobald sie im Netzwerk registriert sind, nicht automatisch entfernt werden, und somit dieser Angriff nicht funktionieren würde. Somit wird der Test angepasst, und nun kann untersucht werden, wie viele Angreifer es braucht, um die Knoten aktiv zu entfernen.

Dieser Test wird ab hier mit **T2 – Legitime Knoten entfernen** gekennzeichnet.

Momentan kann ein Angreifer mit Leichtigkeit alle Informationen sammeln, die er für das Entfernen eines anderen Knoten braucht. Allerdings muss auch hier schlussendlich abgestimmt werden, ob der Knoten entfernt wird oder nicht.

Für diesen Test werden zuerst alle Knoten ausgeschaltet, und dann nacheinander wieder eingeschaltet. Dabei wird für jeden neu eingeschalteten Knoten getestet, ob die Abstimmung, also das Entfernen, schon erfolgreich ist, siehe Abbildung (20). Ähnlich wie beim vorherigen Test kann somit geprüft werden, wie viele Knoten für eine Abstimmung eingeschaltet sein müssen. Die Abstimmung betrifft nun aber eine Entfernung, im Vergleich zu einem Ping. Es wird allerdings nur so lange getestet, bis zum ersten Mal ein Knoten entfernt werden kann. Dies, weil bei der ersten möglichen Entfernung das Verhältnis am meisten gegen den Angreifer spricht.

Als Beispiel, wenn bei 11 Angreifer-Knoten von 15 Knoten insgesamt ein legitimer Knoten entfernt werden kann, also einer der 4 ausgeschalteten, ist das Verhältnis dazu 11 zu 4. Mit demselben Netzwerk ist dann für die nächste Entfernung das Verhältnis 11 zu 3, also vorteilhafter für die Angreifer. Versucht der Angreifer einen Knoten zu entfernen mit 12 Angreifer-Knoten von 15 Knoten insgesamt, ist auch hier das Verhältnis mit 12 zu 3 erneut besser für die Angreifer, im Vergleich zu 11 zu 4, und es muss nicht getestet werden.

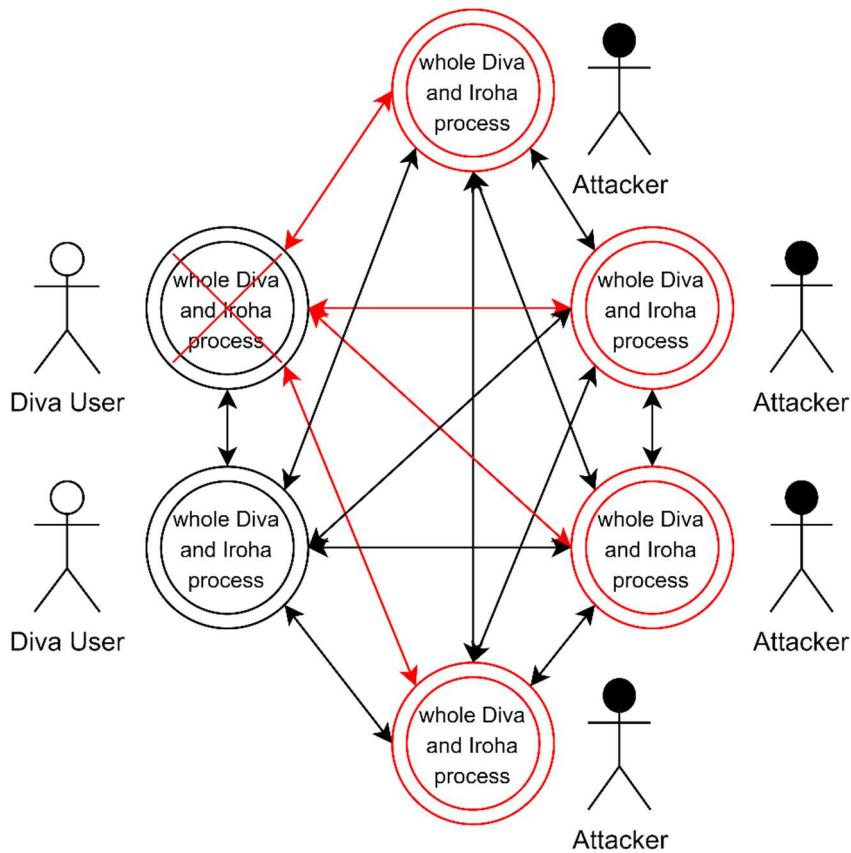


Abbildung 20: Legitime Knoten aktiv entfernen

3.3.2.1 Erwartetes Resultat

Genau gleich wie beim vorherigen Test, müssten auch hier mehr als zwei Drittel einer Entfernung zustimmen. Also wie oben beschrieben, müssten mindestens 11 Knoten von 15 Knoten insgesamt einer Entfernung zustimmen, um die 3f+1 Fehlertoleranz zu erfüllen.

3.3.3 Test T3 – Blockchain manipulieren mithilfe gewisser Mehrheit

Konzeptionell ist dieser Angriff sehr ähnlich zum ersten Angriff. Im Gegensatz zur "Aufnahme von neuen Blöcken blockieren", ist bei diesem Test das Ziel, so viele Knoten zu kontrollieren, um beliebige Blöcke in die Blockchain aufnehmen zu können, siehe (21). Das bedeutet, bei jeder Abstimmung diese Mehrheit zu besitzen, um den Block eigenständig, also ohne Hilfe von legitimen Knoten, aufzunehmen.

Dazu werden, wie beim zweiten Test, zuerst alle Knoten ausgeschalten, und danach wieder einer nach dem anderen eingeschalten. Im Vergleich zum Test 2 wird allerdings nicht versucht Knoten zu entfernen, sondern notiert, ob ein Ping in die Blockchain aufgenommen werden kann. Wie beim ersten Test kann mit einer Aufnahme eines Pings gezeigt werden, dass die Angreifer nun genügend Knoten kontrollieren. Im Gegensatz dazu kann mit einer "nicht-Aufnahme" eines Pings gezeigt werden, dass noch nicht genügend Blöcke kontrolliert werden. Eine Aufnahme sowie eine "nicht-Aufnahme" werden notiert.

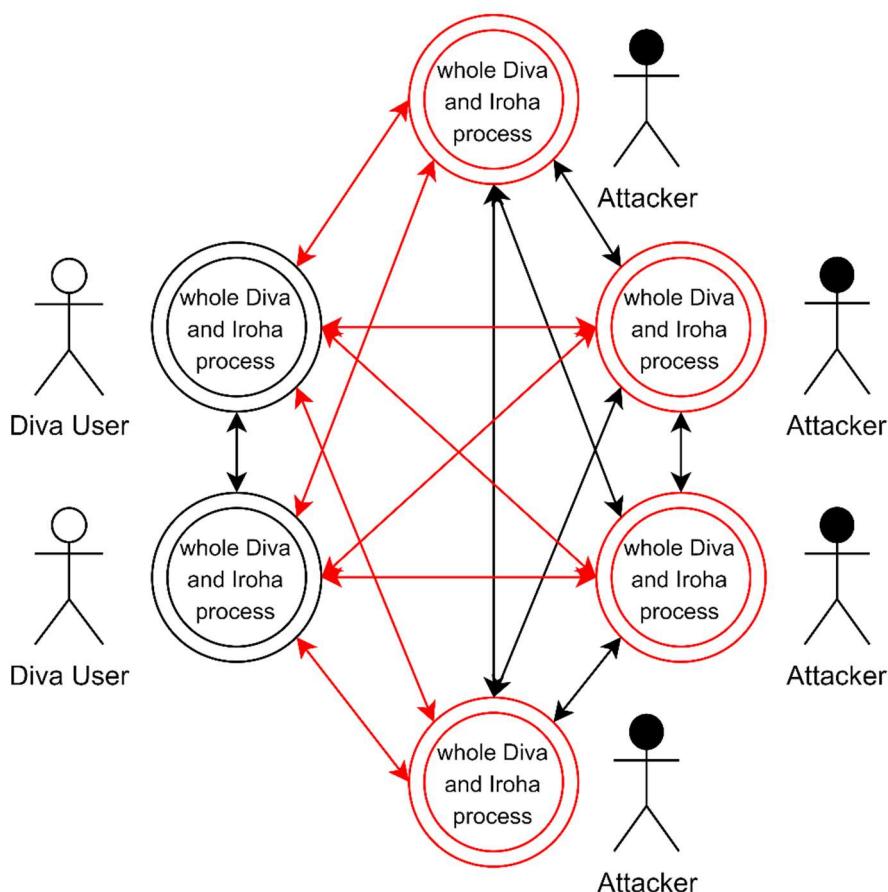


Abbildung 21: Angriff mithilfe gewisser Mehrheit

3.3.3.1 Erwartetes Resultat

Erneut sollte resultieren, dass mehr als zwei Drittel einer Aufnahme eines Pings zustimmen müssen, bevor dieser in die Blockchain aufgenommen wird. Also sollten bei einer Netzwerkgröße von 15 Knoten mindestens 11 Knoten eingeschalten sein, und auch mindestens 11 Knoten den Ping mit ihrer Unterschrift validieren.

3.3.4 Netzwerkgrößen für Tests

Bei allen drei Tests wird jeweils mit 9, 15, 21, 27 und 33 Knoten getestet. Mit diesen Zahlen ist die Differenz zwischen der $2f+1$ und $3f+1$ Fehlertoleranz jeweils am grössten, und gleichzeitig die Gesamtanzahl Knoten minimal. Die Herleitung dieser Zahlen ist im Anhang unter "Differenz $2f+1$ zu $3f+1$ " beschrieben.

Als Beispiel: $2f+1$ bedeutet, dass mehr als die Hälfte der Knoten bei einer Abstimmung zusagen müssen. Bei insgesamt 9 Knoten müssen somit 5 Knoten einem neuen Block zustimmen. $3f+1$ bedeutet, dass mehr als zwei Drittel zustimmen müssen, also 7 Knoten bei insgesamt 9 Knoten. Das bedeutet, dass bei 9 Knoten eine Differenz von 2 Knoten zwischen der $2f+1$ und $3f+1$ Fehlertoleranz herrscht. Bei 15 Knoten insgesamt ist es eine Differenz von 3 Knoten, bei 21 sind es 4, bei 27 sind es 5 und bei 33 sind es 6. Damit soll gezeigt werden, ob Iroha die $2f+1$ oder $3f+1$ Fehlertoleranz implementiert hat.

Mehr als 33 Knoten werden nicht in Betracht bezogen, weil das Netzwerk mit schon 50 Knoten nur noch sehr langsam startet, und die Tests somit viel länger dauern würden und die Resultate unzuverlässiger sein können. Weniger als 9 Knoten werden nicht in Betracht gezogen, weil sonst die Differenz nur noch einen Knoten beträgt, was die Aussagekraft des Tests zunichthemacht.

4 Resultate

4.1 Statische Analyse

In diesem Kapitel werden jene Schwachstellen bewertet, die mithilfe statischer Analyse untersucht und bewertet werden können.

4.1.1 S1 – Verwendete Verschlüsselung und deren Implementation

Für diesen Aspekt wird die Funktionsweise eines Pings bis in die Iroha Library und ihre verwendeten Libraries untersucht. Dafür wird die Methode, die vom Ping aufgerufen wird, aufgeschrieben. Folgenden werden die Methoden notiert, die in der vorherigen aufgerufen werden, und so weiter, bis schlussendlich der gesamte sogenannte Call Stack (22)(23) erstellt ist. Auf den nachfolgenden Seiten ist dieser abgebildet. Das Original ist unter folgendem Link einzusehen:

https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/research/api_investigation.md#calls

Im nachfolgenden Call Stack ist ersichtlich, dass für das Signieren schlussendlich die Library "ed25519.js" und für das Hashen die Library "js-sha3" verwendet wird.

"ed25519" steht für EdDSA mit der Kurve 25519. Dieser Algorithmus ist weit verbreitet und es sind keine theoretischen Schwachstellen bekannt, solange die Implementation korrekt ist [38]. Auch die Implementation besitzt nach Snyk keine Schwachstellen [39].

"js-sha3" bezeichnet eine Implementation des SHA-3 Hash Algorithmus in JavaScript. SHA-3 ist die neueste Version der "Secure Hash Algorithm" Gruppe und es sind ebenfalls keine theoretischen Schwachstellen bekannt [40]. Die von Iroha verwendete Library enthält ebenfalls keine Schwachstelle gemäss Snyk [41].

Somit ist nach öffentlichem Wissen das **Signieren und Hashen von Transaktionen und Blöcken bei Iroha sicher**.

ping call stack

```
in _initIroha(), in /app/src/diva-api.js on line 244
DivaApi._timeout('pingBlockchain', () => DivaApi._pingBlockchain(), MS_1M)
```

getting called from ...

- `_initIroha()`
function on line 223 in /app/src/diva-api.js
getting called from `make()`
 - `make()`
function on line 73 in /app/src/diva-api.js
getting called from `main.js` on line 31
 - `main.js`
script at /app/main.js
getting called from `entrypoint.sh`
 - `entrypoint.sh`
script at /
getting called from Dockerfile
 - `Dockerfile`
script at /
is executed when `docker up`

variable definitions

- `DivaApi`
is the class (this)
- `DivaApi.iroha`
defined in `_init_Iroha()`
`DivaApi.iroha = await Iroha.make(DivaApi.torii, DivaApi.creator, DivaApi.pathIroha)`
- `Iroha.make()`
in /app/src/iroha.js
creates new Iroha object

calls ...

- `_timeout(...)`
function on line 454 in /app/src/diva-api.js
TL;DR fancy wrapper for setting a timeout, then calling the function
 - `_pingBlockchain()`
function on line 438 in /app/src/diva-api.js
calls `DivaApi.iroha.setAccountDetail(DivaApi.creator, 'ping', Math.floor(Date.now() / 1000))`
 - `setAccountDetail(...)`
function on line 242 in /app/src/iroha.js
calls `return commands.setAccountDetail({...}, {...})`
`import { commands, queries } from 'iroha-helpers'`
 - `commands`
defines exports for iroha-helpers
overview [iroha-helpers](#)
`npm pack iroha-helpers` to download source
- in /package/lib/index.js (this is the entrypoint)
`var tslib_1 = require("tslib");`
`var commands_1 = tslib_1._importDefault(require("./commands"));`
`exports.commands = commands_1["default"];`
 (tslib is just a fancy wrapper of ts import functionality)

Abbildung 22: Ping Call Stack (1/2)

weiter auf der nächsten Seite...

```

■ commands.setAccountDetail(...)
export on line 287 from /package/lib/commands
exports setAccountDetail, this means _pingBlockchain() -> iroha.setAccountDetail(...) (from irohajs) ->
setAccountDetail(...) (from /package/lib/commands/index.js)
calls command from /package/lib/commands/index.js

■ command
funtion on line 20 from /package/lib/commands/index.js
takes given options or uses default
calls util_1.signWithArrayOfKeys(...)
calls util_1.sendTransaction(...)
util_1 = require("../util")

■ util.signWithArrayOfKeys(...)
function on line 126 from /package/lib/util.js
signWithArrayOfKeys(...) {... txHelper_1["default"].sign(...)} ...

■ txHelper_1
var on line 5 from /package/lib/util.js
txHelper_1 = tslib_1.__importDefault(require("./txHelper"));

■ txHelper.js
exports sign for util.signWithArrayOfKeys on line 145

■ sign(...)
function on line 99 in /package/lib/txHelper.js
calls hash(...)
calls cryptoHelper_1["default"].sign(...)
gets private key from "buffer_1.Buffer.frompublic

■ hash(transaction)
function on line 91 in /package/lib/txHelper.js
hashes transaction.getPayload() with js_sha3_1.sha3_256

■ js_sha3_1
variable defined in /package/lib/txHelper.js on line 5
var js_sha3_1 = require("js-sha3")

■ js_sha3_1
variable defined in /package/lib/txHelper.js on line 5
var js_sha3_1 = require("js-sha3")

■ js-sha3 in package.json
dependency defined on line 41 in package.json
"js-sha3": "^0.8.0"
no vulns according to https://snyk.io/vuln/npm:js-sha3

■ cryptoHelper_1
variable defined in /package/lib/txHelper.js on line 12
var cryptoHelper_1 = tslib_1.__importDefault(require("./cryptoHelper"))

■ cryptoHelper.js
sign(...) function on line 61, signs the payload with private and public key
ed25519sha3.sign(...) on line 64

■ ed25519sha3
variable defined in /package/lib/cryptoHelper.js on line 5
var ed25519sha3 = tslib_1.__importStar(require("ed25519.js"))

■ ed25519sha3
variable defined in /package/lib/cryptoHelper.js on line 5
var ed25519sha3 = tslib_1.__importStar(require("ed25519.js"))

■ ed25519.js in package.json
dependency defined on line 39 in package.json
"ed25519.js": "1.3.0"
no vulns according to https://snyk.io/vuln/npm:ed25519.js

■ util_1.sendTransaction(...)
function on line 105 from /package/lib/util.js
returns a Promise after _listToTorii(...) has resolved

■ _listToTorii(...)
function on line 7 from /package/lib/util.js
calls hash(...) from txHelper.js (see above)

```

Abbildung 23: Ping Call Stack (2/2)

4.1.2 S2 – Zugriffsrechte auf lokale Schlüssel

Für diese potenzielle Schwachstelle wird untersucht, welche Rechte nötig sind, um auf die lokalen Schlüssel von Iroha zuzugreifen. Dabei stellt sich heraus, dass alle Dateien im Iroha Container die folgenden Zugriffsrechte haben:

```
-rw-rw-r-- 1 root root
```

Das bedeutet, dass der root User des Docker Containers diese Dateien besitzt. Er selbst und alle Benutzer in der root Gruppe können die Schlüssel lesen und anpassen. Die restlichen Benutzer können die Schlüssel lesen. Normalerweise wären diese Rechte unsicher, weil jeder Benutzer des PCs die Dateien und somit den privaten Schlüssel lesen könnte. Allerdings können diese Benutzer nicht darauf zugreifen, weil es sich um einen Docker Container handelt und dieser root Rechte, oder zumindest Rechte auf den Docker Prozess benötigt, also den "containerd" Prozess. Siehe dazu das Skript "inspect.sh" im Abschnitt "Docker Zugriffsrechte" im Anhang.

Mit dem Skript "inspect.sh" kann gezeigt werden, was geschieht, wenn ein normaler Benutzer und der root User versucht, die Schlüssel zu lesen. Beim root User ist ersichtlich, dass die Schlüssel normal gelesen werden können. Versucht ein normaler Benutzer die Schlüssel über das Filesystem zu lesen, erhält er die Meldung "permission denied", also "Zugriff verweigert". Versucht er es über den Docker Prozess, also das Docker für ihn die Schlüssel liest, erhält er erneut "permission denied", weil er keine Rechte hat, dem Docker Prozess Befehle zu erteilen.

Ist aber ein System falsch konfiguriert, oder es besitzt eine Schwachstelle, sodass sich normale Benutzer unautorisiert root Rechte beschaffen können (privilege escalation), kann damit auch oben genannte Sicherheitsvorkehrung umgangen werden. DIVA.EXCHANGE und Docker machen jedoch alles ihnen Mögliche, um diese Schwachstelle zu schliessen.

Mithilfe dieser theoretischen und praktischen Analysen kann gezeigt werden, dass **die Schlüssel in den Docker Container nur für den Docker Container selbst oder den root User des Systems lesbar sind.**

4.1.3 S3 – Zugriffsrechte und Anpassung der lokalen Blockchain

Die Blöcke der lokalen Blockchain ist im Ordner "blockstore" ersichtlich. Die Datei 00...01 bildet dabei den ersten Block der Blockchain ab. Wie im vorherigen Abschnitt S2 hat diese Datei die Zugriffsrechte:

```
-rw-rw-r-- 1 root root
```

Mit derselben Logik wie im vorherigen Abschnitt kann gezeigt werden, dass die lokale Blockchain nur vom Docker Prozess und dem root User angepasst werden. Um zu garantieren, dass die Blockchain nicht ungewollt angepasst wird, muss auch noch gezeigt werden, dass die Blockchain nur erweitert werden kann. Alte Stände dürfen durch die API nicht beliebig anpassbar sein. Dazu muss die gesamte Iroha-API und Diva-API untersucht werden.

In der Diva-API sind keine verdächtigen Funktionen vorhanden, die auf solche Funktionalität hinweisen. Auch das Entwicklerteam von DIVA.EXCHANGE ist sich einer solchen Funktion nicht bewusst. Die Iroha-API ist nur beim Ping genau untersucht worden. DIVA.EXCHANGE plant, ihre eigene Blockchain zu implementieren, und somit ist es sinnvoller bei dieser **neuen Blockchain von DIVA.EXCHANGE eine genauere Untersuchung der Funktionsweise** durchzuführen, und nicht bei der Iroha-API.

4.2 Praktische Analyse – Teststand

In diesem Kapitel werden die Resultate der Tests mithilfe von Diagrammen dargestellt. Die Daten der Diagramme stammen direkt von den Testresultaten, welche im Anhang zu finden sind. Auf den folgenden Seiten sind diese Resultate auf insgesamt neun Graphen abgebildet. Die Methodik der Tests, die möglichen Auswirkungen und die Erwartungswerte sind im Kapitel "3.3 Tests" detaillierter beschrieben.

4.2.1 Übersicht & Evaluation

Für jeden der drei Tests werden zuerst je drei Diagramme gezeigt. Das erste Diagramm zeigt, wie viele Knoten es für eine Abstimmung braucht, also $3f+1$ oder eben $2f+1$. Das zweite zeigt, wie viele Knoten jeweils an einer erfolgreichen Abstimmung teilgenommen haben. Das dritte zeigt, wie lange Iroha benötigt, um für eine Abstimmung einen Konsens zu finden.

Bei jedem Diagramm werden Erkenntnisse festgehalten. Diese Erkenntnisse werden dann im Kapitel 5 zusammengefasst, über ihre Auswirkungen diskutiert, und welche Massnahmen DIVA.EXCHANGE treffen soll, um ihr Produkt sicherer zu gestalten.

4.2.1.1 T1 - Neue Einträge in die Blockchain verhindern

Die Tabellen zu den nachstehenden Diagrammen sind im Anhang unter "Resultate – Tabellen" einzusehen.



Abbildung 24: T1 - Ping vs. Kein Ping

Im obenstehenden Diagramm (24) ist ersichtlich, wie viele Knoten einem Ping zustimmen müssen, damit dieser in die Blockchain niedergeschrieben wird. Wie erwähnt, sind nur die Netzwerkgrößen mit 9, 15, 21, 27 und 33 Knoten getestet, weil da die Differenz zwischen $2f+1$ und $3f+1$ am grössten ist.

Die roten Karos bilden ab, dass mit einer gewissen Anzahl von abstimmenden Knoten (Y-Achse) bei einer gegebenen Netzwerkgröße (X-Achse) kein neuer Ping in die Blockchain aufgenommen wird. Die grünen Knoten deuten darauf hin, dass der Ping aufgenommen wird. Die dunkelviolette Linie zieht die Grenze zu $3f+1$. Über dieser Linie ist $3f+1$ erfüllt, also mehr als zwei Drittel stimmen einem Ping zu. Die hellviolette Linie stellt $2f+1$ dar. Über dieser Linie stimmen mehr als die Hälfte der Knoten einem Ping zu.

Wie unschwer zu erkennen ist, **erfüllt Iroha nur die Vorgaben des $2f+1$, und nicht die des $3f+1$.** Bei 9 und 15 Knoten wird der Ping noch beim absoluten Minimum für $2f+1$ angenommen, bei 21 hätte ein Ping zusätzlich angenommen werden können, und bei 27 und 33 Knoten hätten nochmals 3 zusätzlich angenommen werden können. Das wird allerdings am ehesten mit der Performance von Iroha zusammenhängen, wie auf den folgenden zwei Seiten gezeigt wird.

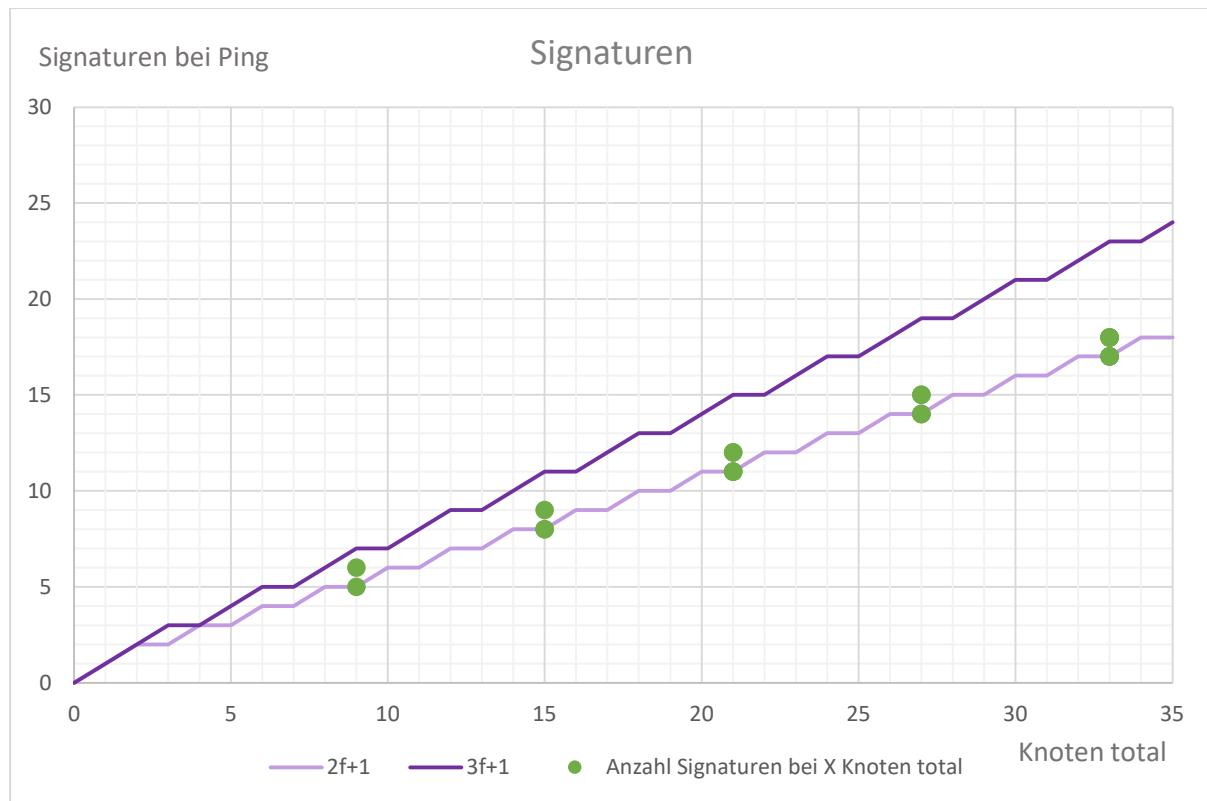


Abbildung 25: T1 - Signaturen

In diesem Diagramm (25) ist dargestellt, wie viele Knoten ein Ping validieren. Im Vergleich zur vorherigen Seite wird also nicht geschaut, wie viele Knoten aktiv abstimmen müssen, sondern wie viele Knoten, die abstimmen können, auch wirklich abstimmen. Die violetten Linien beschreiben erneut die $2f+1$ und $3f+1$ Grenzen.

Als Beispiel, auf der vorletzten Seite ist die Tabelle mit einer Netzwerkgrösse von 33 Knoten abgebildet. In der ersten Zeile ist ersichtlich, dass alle 33 Knoten abstimmen könnten, also alle sind eingeschalten, trotzdem validieren nur 17 Knoten einen neuen Ping. Weil in einem lokalen Netzwerk getestet wird, ist die Latenz sehr klein und die Knoten benötigen nur wenige Millisekunden, um miteinander zu kommunizieren. Das führt vermutlich dazu, dass manchmal ein zusätzlicher Knoten den Ping validiert, weil die extra Signatur so schnell eintrifft. In einem verteilten, nicht lokalen Netzwerk sollte das nicht auftreten. Und auch wenn es auftritt, würde dies Iroha nicht sicherer machen, da die Schwankungen so klein sind, dass auch damit $3f+1$ nie erfüllt worden ist.

Schlussendlich ist auch hiermit gut ersichtlich, dass **Iroha die Fehlertoleranz $2f+1$ implementiert**. Wäre $3f+1$ implementiert, müssten alle grünen Punkte auf oder über der dunkelvioletten Linie verharren. Die gesammelten Daten passen zu perfekt mit $2f+1$ zusammen.

Theoretisch wäre es möglich, dass nicht mal $2f+1$ erfüllt sein müsste. Aber bei allen Tests kam es noch nie vor, dass nur die Hälfte oder weniger zustimmte. Mit allen Prototypen und Probefahrten summieren sich die Pings, die aufgenommen oder abgewiesen worden sind, sehr wahrscheinlich auf über 1000 Pings. Und wenn angenommen wird, dass die Chance bei jedem Ping 50/50 war, ob $2f+1$ durch Zufall erfüllt wird oder nicht, wäre die Wahrscheinlichkeit, dass nicht mal $2f+1$ implementiert ist, bei 1 zu 2^{1000} , also 1 zu 10^{301} . Somit ist es praktisch unmöglich, dass Iroha mit noch weniger als $2f+1$ implementiert ist. Wäre die Chance mehr bei 1 zu 10^{20} , wäre es gerade noch möglich [42].

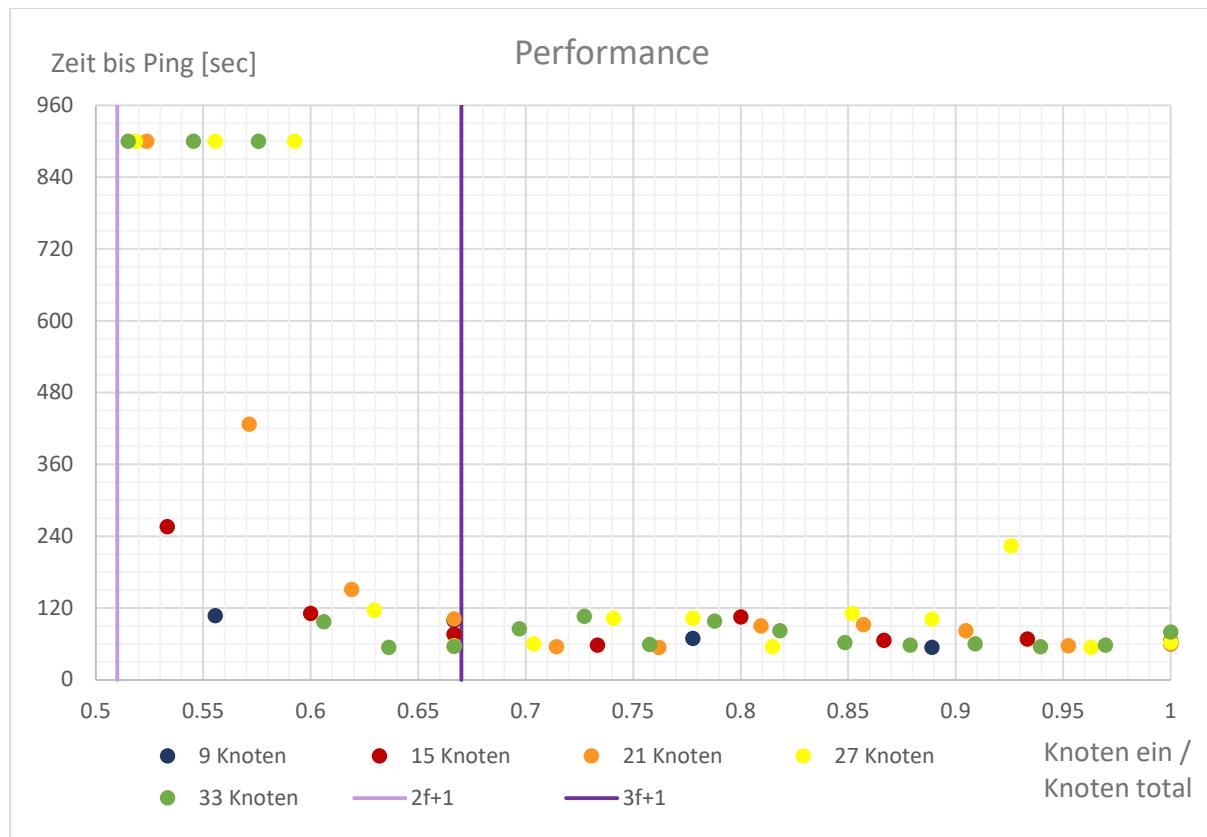


Abbildung 26: T1 - Performance

Im Diagramm oberhalb (26) wird festgehalten, wie sich die Performance von Iroha verhält, wenn nur ein gewisser Teil der Knoten abstimmt. Wie auf der vorletzten Seite erwähnt, fehlt bei der Netzwerkgrösse von 21 Knoten ein Ping, und bei den Grössen von 27 und 33 sogar drei.

Auf der X-Achse ist das Verhältnis zwischen Knoten, die abstimmen und Knoten insgesamt festgehalten. Die Y-Achse zeigt, wie lange es dauert, bis bei einem gewissen Verhältnis ein Ping aufgenommen wird. Eigentlich müsste alle 60 Sekunden ein Ping aufgenommen werden. Bei einem Verhältnis von mehr als 0.6 werden alle Pings zwischen 60 und 120 Sekunden niedergeschrieben, nur eine Ausnahme gibt's dazu. Als Beispiel, sind 9 Knoten noch am Abstimmen von 15 Knoten, ist das ein Verhältnis von 9 zu 15 oder eben 0.6.

Nähert sich das Verhältnis allerdings der 0.5 Grenze, geht die Performance zu Boden und die Zeit, einen Ping anzunehmen, vervielfacht sich. Somit ist ersichtlich, dass die Laufzeit des Algorithmus abhängig vom Verhältnis zwischen antwortenden Knoten und der Netzwerkgrösse.

Die Erklärung dafür ist wie folgt: Bei einem neuen Ping wird ein Knoten bestimmt, der die ganze Abstimmung organisieren soll. In den Tests ist das aller meistens der Knoten 1, weil darauf die Diva-API läuft und der Ping davon stammt, und somit Knoten eins die kleinste Latenz besitzt. Dieser Knoten bestimmt dann mithilfe eines vorgeschriebenen Algorithmus eine Menge an Knoten, und wartet dann nacheinander auf dessen Antwort. Kommt von einem Knoten keine Antwort, wird angenommen, dass der Algorithmus nochmals eine neue Menge auswählt, und erneut auf alle Antworten wartet. Das ist sehr ineffizient und schlecht konzipiert, weil bei gewissen Bedingungen die Laufzeit exponentiell zunehmen kann. Ein Rechenbeispiel dazu folgt auf der nächsten Seite. Die Rechnungen sind mit dem Skript "correct_shuffle_chance.py" durchgeführt worden. Dieses Skript ist im Anhang unter "Wahrscheinlichkeit für korrekte Auswahl von Abstimmungsknoten" genauer beschrieben.

Ein Beispiel des Skripts mit 33 Knoten: Sind nun nur noch 17 Knoten eingeschalten, muss der Organisator der Abstimmung genau die 16 anderen eingeschalteten Knoten erwischen, er selbst stimmt ja auch ab. Dazu sind seine Chancen für den ersten Knoten bei 16/32, danach 15/31, dann 14/30, ..., bis schlussendlich 1/17, um den letzten eingeschalteten Knoten zu erwischen. Das entspricht einer Chance von 1 zu 601 Millionen, also 601 Millionen Mal müsste Iroha merken, dass die Knoten nicht antworten. Und das nur für einen einzigen Ping. In einem verteilten Netzwerk, wo die Latenz auch mal bis zu einer Sekunde dauern kann, wären das im Durchschnitt etwa 9.5 Jahre, bis eine korrekte Verteilung gefunden wird, bei 17 eingeschalteten Knoten von 33 Knoten insgesamt.

Als weiteres Beispiel wird ein Messpunkt des vorherigen Graphs genommen, den orangen Punkt bei 0.57 und 420 Sekunden. Hier gibt es eine korrekte Menge von etwa 168k möglichen. Bei 420 Sekunden schreibt das eine durchschnittliche Latenz von 5ms vor, damit Iroha die Abwesenheit eines Knoten erfassen kann und den nächsten Shuffle durchführen, was bei einem lokalen Netzwerk gut möglich wäre.

Ob der Algorithmus exakt so funktioniert, konnte in der gegebenen Zeit nicht bestätigt werden. Mit Sicherheit kann jedoch festhalten werden, dass der **Algorithmus von Iroha funktionieren kann bei Netzwerken über 100 Knoten**. Auch trotz dem Performance-Einbruch wird die **Sicherheit nicht auf das Level von 3f+1 gehoben**, weil zum einen schon bei 0.6 das Netzwerk mehr oder weniger funktioniert anstatt bei erst 0.67 (3f+1), und zum anderen eine solche Methode keine Garantien zur Sicherheit des Systems geben kann.

Zusätzlich ist zu beachten, dass ein Angreifer seinen Angriff optimieren kann, wenn er **die Latenz der anderen Knoten manipuliert** und seine eigenen Knoten die kleinste Latenz zur API haben. So kann er **selbst die Knoten bestimmen, die abstimmen werden**, anstatt Zufällige Knoten auszuwählen. Er bräuchte trotzdem eine 2f+1 Mehrheit, aber die Perfomance würde aufrecht erhalten bleiben, weil sofort genügend Knoten ihm antworten.

4.2.1.2 T2 - Legitime Knoten entfernen

Die Tabellen zu den nachstehenden Diagrammen sind im Anhang unter "Resultate – Tabellen" einzusehen.

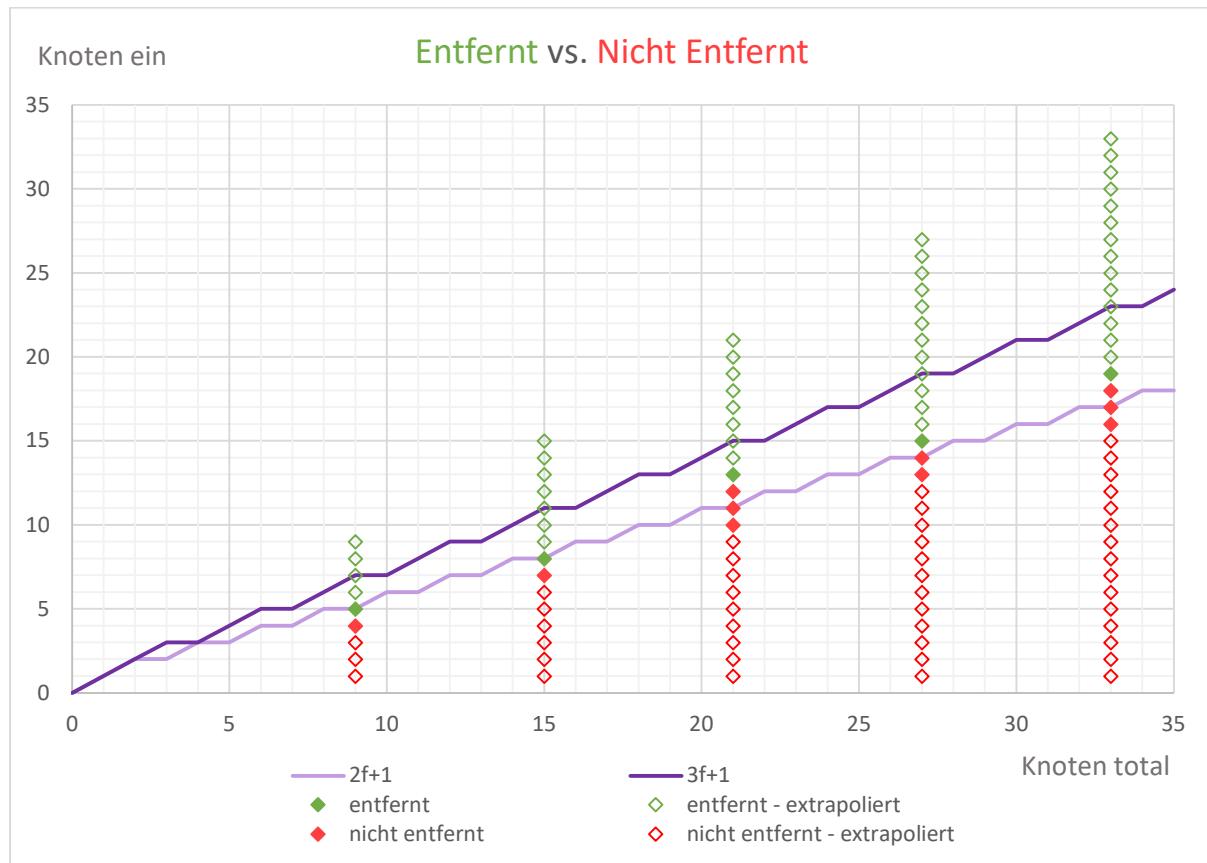


Abbildung 27: T2 - Entfernt vs. Nicht Entfernt

Ähnlich wie beim ersten Test ist die Anzahl der abstimmenden Knoten und die Gesamtanzahl abgebildet (27) mit der Y- respektive der X-Achse. Allerdings wird bei diesem Test notiert, wie viele Knoten einer Entfernung eines Knoten zustimmen müssen.

Folglich bedeuten die roten, gefüllten Karos, dass keine Entfernung stattfinden konnte. Ein grünes, gefülltes Karo heisst, es ist erfolgreich ein Knoten entfernt worden. Die leeren, grünen Karos deuten darauf hin, dass eine Entfernung höchst wahrscheinlich ist, aber dass aus Zeitgründen nicht getestet werden kann. Gleichermaßen mit den leeren, roten Karos, auch diese können aus Zeitgründen nicht gut getestet werden, aber diese sind mit höchster Wahrscheinlichkeit nicht möglich. Die violetten Linien beschreiben wie vorhin die Grenzen von $2f+1$ und $3f+1$.

Mit den Erkenntnissen dieses Graphs und dem Einbezug des nächsten Graphen lässt sich herleiten, dass auch **beim Entfernen nur die Fehlertoleranz $2f+1$ erfüllt werden muss, und nicht $3f+1$** .

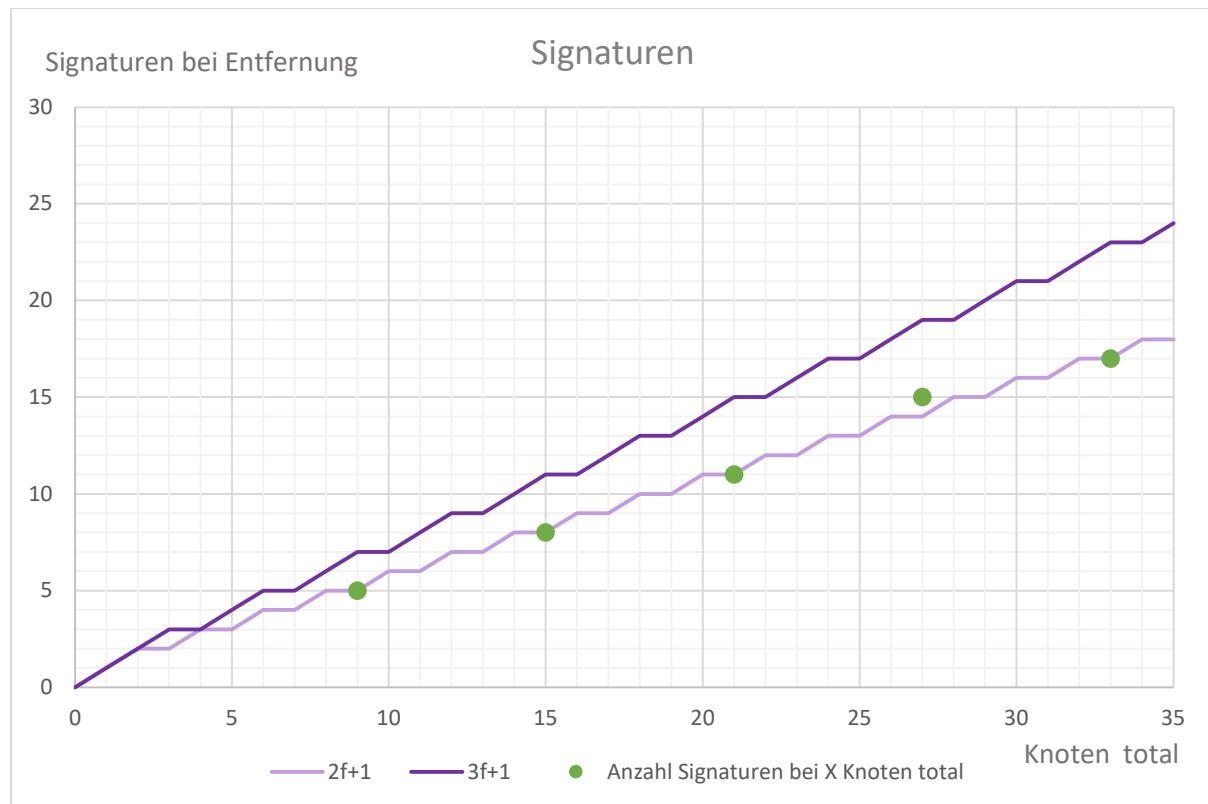


Abbildung 28: T2 - Signaturen

Auch dieses Diagramm (28) ist dem aus dem ersten Test sehr ähnlich. Anstatt zu notieren, wie viele Knoten ein Ping validieren, wird hier gezeigt, wie viele Knoten eine Entfernung eines anderen Knoten validieren müssen.

Die grünen Punkte stellen dabei die Anzahl der Signaturen (Y-Achse) bei einer gewissen Netzwerkgrösse (X-Achse) dar. Nicht ersichtlich ist, wie viele Knoten die Entfernung hätten validieren können. Als Beispiel, bei der Netzwerkgrösse von 21 Knoten sind alle 21 Knoten eingeschalten, trotzdem validierten schlussendlich nur 11 Knoten die Entfernung.

Erneut lassen die beobachteten Zahlen nur den Schluss zu, dass **2f+1 anstatt 3f+1 implementiert** sein muss.

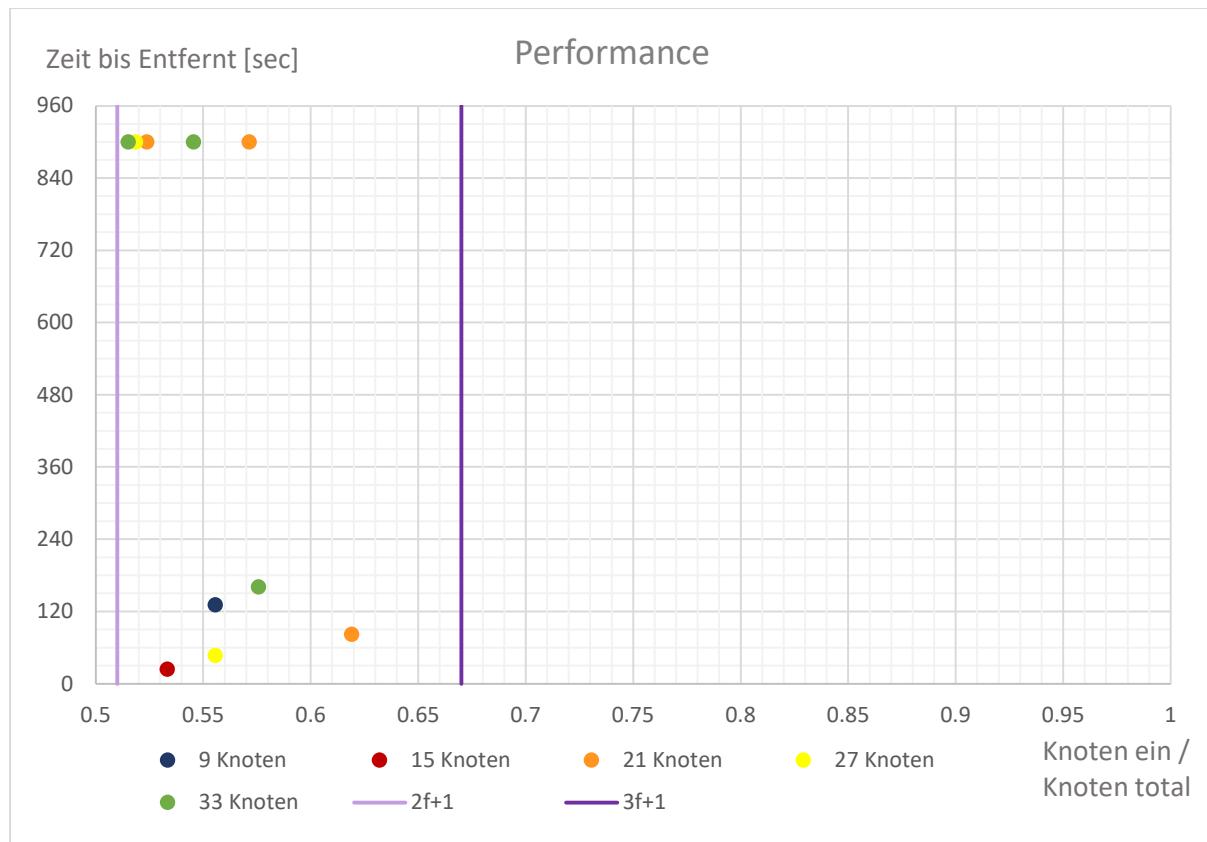


Abbildung 29: T2 - Performance

Aufgrund des gekürzten Tests sind hier (29) nur wenige Messpunkte vorhanden. Die gezeigten Messpunkte legen dar, wie lange es dauert, bis ein Knoten entfernt ist. Auf der X-Achse ist das Verhältnis von eingeschalteten Knoten zu Knoten insgesamt abgebildet, auf der Y-Achse die benötigte Zeit.

Erneut ist ersichtlich, dass auch das Entfernen sehr lange dauern kann. Mehr als 900 Sekunden bei einem Netzwerk mit mehr als 20 Knoten und einem Verhältnis von 0.51 bis etwa 0.6. Als Vergleich dazu: Wenn alle Knoten eingeschalten sind, dauert es nur 5 Sekunden für ein Netzwerk mit 9 Knoten, und 7 Sekunden für ein Netzwerk mit 33 Knoten, siehe "Dauer für eine Entfernung bei einem vollständig aktiven Netzwerk" im Anhang.

Auch hier lässt sich nur der Schluss ziehen, dass **der Algorithmus, der die validierenden Knoten auswählt, nicht funktioniert für ein solches Szenario**. Obwohl es lange dauert, und das Feature "Knoten entfernen" aus dem Code der Diva-API entfernt wird, bietet die aktuell implementierte Funktion keinen $3f+1$ Schutz, sondern nur $2f+1$. Zudem kann jeder Knoten **eine Entfernung** eines anderen Knoten anfordern, und die normale Verhaltensweise der Knoten ist es, diese Anfrage **anzunehmen**, was das ganze System im Moment **sehr unsicher und anfällig für solche Angriffe macht**.

4.2.1.3 T3 – Blockchain manipulieren mithilfe gewisser Mehrheit

Die Tabellen zu den nachstehenden Diagrammen sind im Anhang unter "Resultate – Tabellen" einzusehen.

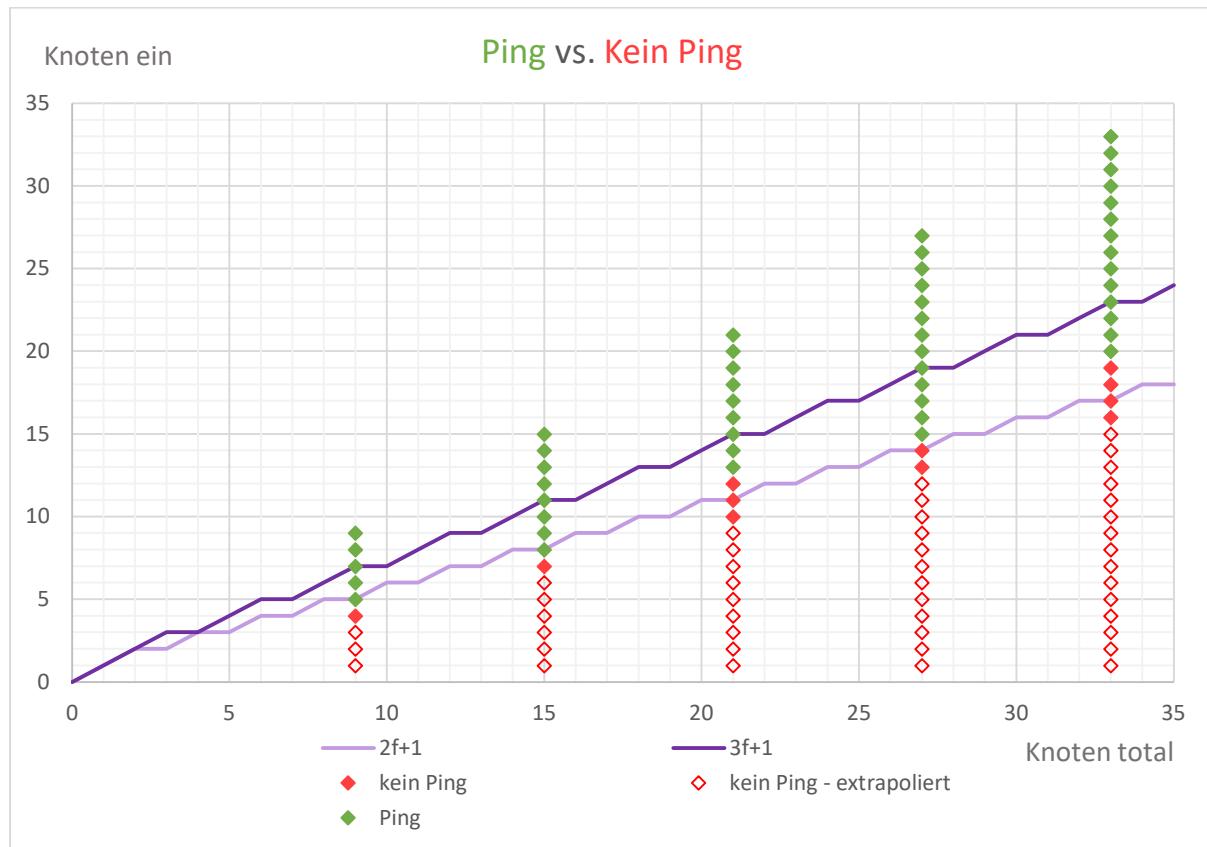


Abbildung 30: T3 - Ping vs. Kein Ping

Konzeptionell deckt dieser Test einen anderen Aspekt ab, im Vergleich zum ersten Test, allerdings ist die Implementierung sehr ähnlich. Auch hier (30) wird notiert, bei wie vielen eingeschalteten Knoten ein Ping in die Blockchain aufgenommen werden kann. Allerdings werden die Knoten nicht nacheinander aus-, sondern nacheinander eingeschaltet und auf einen Ping gewartet. Die Reihenfolge davon ist in den Diagrammen nicht mehr ersichtlich, womit die folgenden drei Diagrammen den dreien des ersten Tests sehr ähnlich sind.

Die roten Karos bilden ab, dass mit einer gewissen Anzahl von abstimgenden Knoten (Y-Achse) bei einer gegebenen Netzwerkgrösse (X-Achse) kein neuer Ping in die Blockchain aufgenommen wird. Die grünen Knoten deuten darauf hin, dass der Ping aufgenommen wird. Ein leeres, rotes Karo bedeutet, wie im zweiten Test, dass diese Konstellation aus Zeitgründen nicht getestet wird, aber das Resultat mit allerhöchster Wahrscheinlichkeit ein fehlender Ping wäre. Die violetten Linie zieht wie gewohnt die Grenze zu $2f+1$ und $3f+1$.

Erneut ist leicht zu erkennen, dass auch hier **Iroha nur $2f+1$ erfüllt, und nicht $3f+1$** . Somit ist die Anzahl, um ein Netzwerk komplett zu kontrollieren genau gleich, wie die Anzahl ein Netzwerk zu blockieren, also in beiden Fällen eine einfache Mehrheit. Wäre $3f+1$ implementiert, bräuchten die Angreifer zwar nur einen Drittel, um das Netzwerk zu blockieren, jedoch mehr als zwei Drittel, um das Netzwerk zu kontrollieren. Im Kapitel "5 Diskussion" wird genauer auf diesen Aspekt eingegangen.

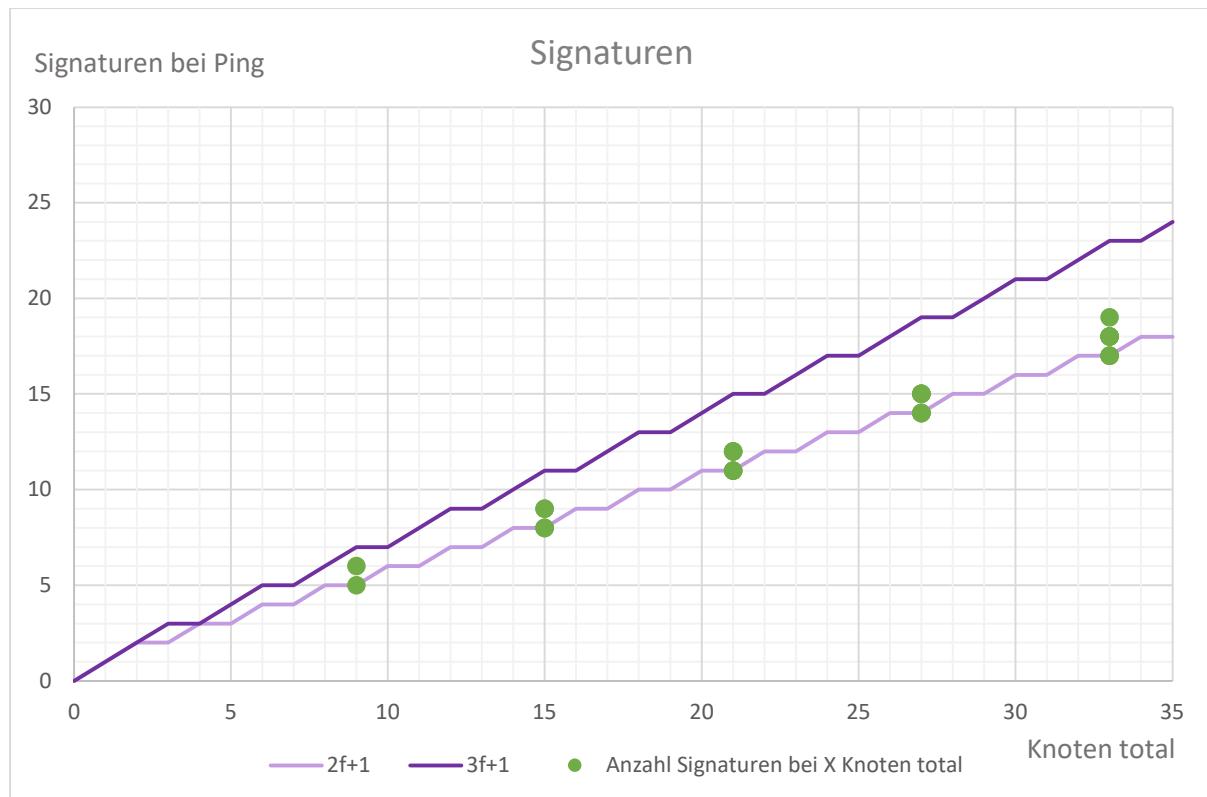


Abbildung 31: T3 - Signaturen

Erneut ist das Resultat fast identisch wie das des ersten Tests. Nochmals wird gezeigt (31), wie viele eingeschaltete Knoten einen Ping validieren. Auch hier gilt, die Anzahl der Signaturen ist nicht unbedingt gleich wie die Anzahl der eingeschalteten Knoten.

Die Anzahl der validierenden Knoten ist mit einem grünen Punkt dargestellt, die Grenzen zu $2f+1$ und $3f+1$ sind wieder violette Linien.

Im Vergleich zum ersten Test haben hier bei einer Netzwerkgrösse von 33 Knoten zwischen 17 und 19 Knoten einen Ping validiert, nicht nur 17 oder 18. Dass erneut bei der Netzwerkgrösse von 33 Knoten 20 Knoten eingeschalten sein müssen, dass aber nur 18 Knoten den Ping validieren, verstärkt erneut die Schlussfolgerung eines **ineffizienten und unpassenden Algorithmus von Iroha für die Auswahl der validierenden Knoten**.

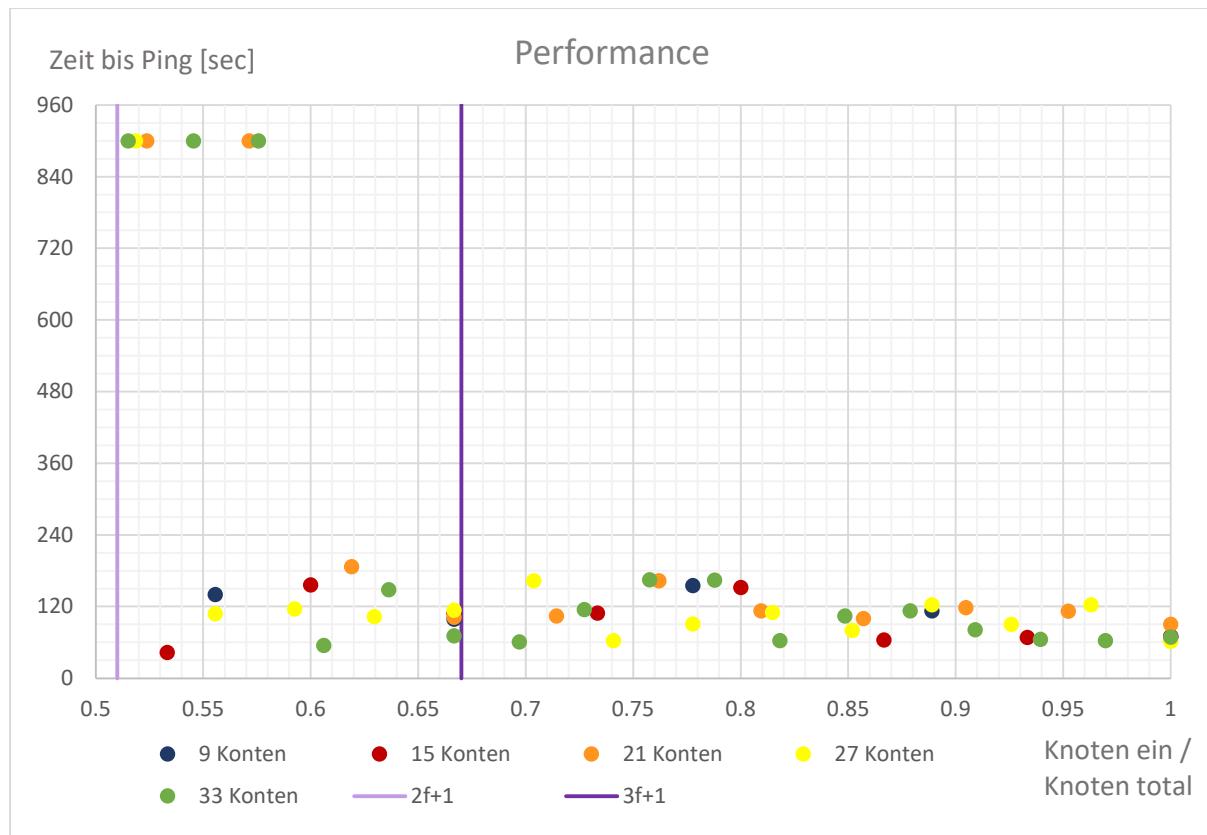


Abbildung 32: T3 - Performance

Auch dieses Diagramm (32) ist dem des ersten Tests sehr ähnlich. Wie bei den letzten zwei Tests zeigt dieses Diagramm auf, wie lange man warten muss, bis ein Ping in die Blockchain aufgenommen wird. Auf der X-Achse ist das Verhältnis von abstimmenden Knoten zu Gesamtanzahl Knoten dargestellt. Die Y-Achse beschreibt in Sekunden, wie lange es dauert, bis der Ping aufgenommen wird.

Wie bei den anderen Tests ist die Performance ab einem Verhältnis von 0.6 einigermassen normal, alles darunter kann in sehr langen Wartezeiten enden.

Weil die Resultate fast dieselben sind wie beim ersten Test, gibt es nur eine neue Erkenntnis. Wenn die Knoten nacheinander ausgeschalten werden, hat das Netzwerk normal funktioniert, bis plötzlich zu wenige eingeschalten sind und keine neuen Blöcke aufgenommen werden. Das heisst alle Pings und sonstige Anfragen sind bis dahin vollständig verarbeitet. Werden die Knoten jedoch nacheinander eingeschalten, bis es funktioniert, sind viele Pings noch nicht verarbeitet. Somit ist das **Netzwerk schon sehr ausgelastet**, bis der erste Ping erscheint, bei dem genügend Knoten eingeschaltet sind.

Wenn der Test allerdings nicht alle Knoten abschaltet, sondern nur die Hälfte, ist das Netzwerk nur mit einem Ping belastet, der noch nicht verarbeitet werden kann, bevor weitere Knoten eingeschaltet werden. Auf die andere Seite kann erneut gezeigt werden, dass Knoten, die über längere Zeit nicht antworten, die Performance des Netzwerks deutlich verschlechtern, und **der Algorithmus von Iroha vollständig ungeeignet ist für eine solche Blockchain**.

4.3 Risikominimierung

Im folgenden Kapitel werden die Aspekte aufgelistet und untersucht, die nicht vollständig analysiert werden können, aber trotzdem eine potenzielle Schwachstelle darstellen, sollten sich die Entwickler von DIVA.EXCHANGE diesen Aspekten nicht bewusst sein.

4.3.1 R1 - Veraltete und unbekannte Verschlüsselungen

Im Moment betrifft dieser Aspekt nur die Entwickler von Iroha, weil DIVA.EXCHANGE noch die Verschlüsselung und Funktionen von Iroha verwendet. Allerdings soll DIVA.EXCHANGE eine eigene Blockchain erhalten, wobei dieser Aspekt von hoher Wichtigkeit ist.

Meistens sind die Argumente für eine eigene Verschlüsselung: "Wie sollen Angreifer diese Verschlüsselung knacken, wenn diese nicht öffentlich ist oder einfach nicht dokumentiert ist?" oder "Andere Verschlüsselungen haben ja auch Schwachstellen."

Zum ersten Punkt: DIVA.EXCHANGE ist Open Source und wird es auch bleiben, was ein Angriff auf eine eigene Verschlüsselung noch einfacher gestalten würde, weil der Source Code ersichtlich ist.

Zum zweiten: Schwachstellen von Verschlüsselungen können vermieden werden, wenn eine öffentlich akzeptierte Library korrekt implementiert wird, sowie es bei Iroha mit js-sha3 und ed25519 der Fall ist, siehe "Statische Analyse".

Sicherheit mithilfe von eigenen Verschlüsselungen kann funktionieren, allerdings darf das Weglassen der Dokumentation der eigenen Verschlüsselung nicht die einzige Sicherheitsmaßnahme sein [43].

4.3.2 R2 - Sichere Passwörter und regelmässige Code Checks

Wie im Kapitel "2.6 Angriffe" erwähnt, kann ein Angreifer auch durch einen Supply Chain Angriff eine Schwachstelle in DIVA.EXCHANGE einbauen und diese dann ausnützen. Codeberg und Docker Hub selbst besitzen keine öffentlich bekannte Schwachstelle, aber ein Angreifer könnte eine Schwachstelle in einer von DIVA.EXCHANGE verwendeten Library suchen und ausnutzen, oder selbst den Code von DIVA.EXCHANGE, Iroha, oder einer verwendeten Library anzupassen versuchen. Grundsätzlich kann jeder zu einem Open Source Projekt beitragen. Auch wenn diese Beiträge meist ordentlich und systematisch überprüft werden, bevor sie in den Code aufgenommen werden, kann es auch bei sehr grossen Projekten wie dem PHP Source Code dazu kommen, dass es Angreifern gelingt, Schwachstellen einzubauen [44].

Somit kann es von grossem Vorteil sein, wenn DIVA.EXCHANGE selbst die verwendeten Libraries bestimmen kann, und nicht auf Iroha angewiesen ist, um neuen Code und verwendete Libraries ordentlich zu analysieren. Es wird empfohlen, den Source Code von DIVA.EXCHANGE regelmäßig zu überprüfen, vor allem die neuen Beiträge von neuen Entwicklern. Zusätzlich sollte eine Meldung aktiviert werden, wenn bei einer verwendeten Library eine Schwachstelle gefunden wird (siehe GitHub und npm). Zudem sollen die DIVA.EXCHANGE-Entwickler in den Aspekten "sichere Passwörter" und "Abwenden von Phishing Angriffen" geschult sein.

5 Diskussion

5.1 Erkenntnisse

Mit Hilfe der Resultate aus Kapitel 4 können folgende sechs Schlüsse zur Sicherheit von DIVA.EXCHANGE und dem Hyperledger Iroha gezogen werden:

1. Es kann mithilfe einer theoretischen und praktischen Analyse gezeigt werden, dass lediglich der root User und der Docker-Container selbst die Schlüssel eines Docker-Blocks auslesen können.
2. Es ist innerhalb der Diva-API keine Funktionen vorhanden, die das Verändern der bereits bestehenden Blockchain ermöglichen. Zusätzlich ist die lokale Blockchain, gleich wie die Schlüssel, maximal vor unautorisierten Anpassungen geschützt. Die Iroha-API könnte unsichere Funktionen bereitstellen, womit die lokale Blockchain angepasst werden könnte. Allerdings plant DIVA.EXCHANGE, ihre eigene Blockchain zu implementieren, womit es für zukünftige Arbeiten sinnvoller ist, diese zu untersuchen.
3. In der statischen Analyse der Libraries wird festgestellt, dass nach öffentlichem Wissen das Signieren und Hashen von Transaktionen und Blöcken bei Hyperledger Iroha sicher ist.
4. In allen drei Tests wird unabhängig voneinander der Schluss gezogen, dass Hyperledger Iroha schon funktioniert, wenn mehr als die Hälfte aller Knoten eines Netzwerkes ordnungsgemäß agieren. Dies entspricht einer byzantinischen Fehlertoleranz von $2f+1$. In keinem der durchgeführten Tests ist der von Iroha versprochene Sicherheitsstandart von $3f+1$ erreicht worden.
5. In allen Tests wird unabhängig voneinander festgestellt, dass die Performance von Hyperledger Iroha stark abnimmt, wenn nur etwas mehr als die Hälfte der Knoten eingeschaltet sind, also nahe bei $2f+1$. Häufig bei einer Netzwerkgrösse von 21 Knoten, spätestens jedoch bei 27 Knoten, kann es über 45 Minuten dauern, bis das Netzwerk fähig ist zu agieren und einen Konsens für neue Blöcke findet. Diese schlechte Performance ist aber kein Argument für einen besseren Schutz gegen $2f+1$, weil das Netzwerk zum einen schon vor $3f+1$ wieder normal funktioniert und zum anderen keine Garantie gegeben werden kann.
 - a. Zusätzlich zeigen die Resultate, dass der Knoten, der die zu abstimgenden Knoten bestimmt, meistens der Knoten n1 ist. Daraus lässt sich schliessen, dass die Abstimmungen immer von dem Knoten organisiert wird, der die kleinsten Latenz zur Diva-API hat. Somit kann ein Angreifer, sollte einer seiner Knoten die kleinste Latenz zur Diva-API besitzen, die Abstimmungen ausfallen lassen. Zusätzlich wäre es möglich, schnelle Abstimmungen zu garantieren, auch wenn er nur knapp über die Hälfte der Knoten kontrolliert, im Vergleich zur gezeigten Performance beim Punkt 5. Das alles gilt jedoch nur, wenn wie beim Testnetz alle Abstimmungen von einer zentralen Diva-API stammen. Weil im Normalfall jeder seine eigene Kopie der Diva-API hat, ist ein solcher Angriff in der realen Welt nicht möglich.
6. Im zweiten Test wird gezeigt, dass mit der aktuellen Version der Diva-API jeder Knoten die Entfernung eines anderen Knoten veranlassen kann. Das Standardverhalten der Knoten ist es, diese Entfernung zu bewilligen. Die Knoten werden allerdings nicht automatisch entfernt, wenn sie über längere Zeit inaktiv sind.

5.2 Falsifizieren der Hypothesen

Folgend sind die Hypothesen aufgelistet und werden mithilfe der Erkenntnisse der vorherigen Seite entweder falsifiziert oder nicht.

5.2.1 Hypothese 1

Die Blockchain-Ebene weist alle Blöcke ab, bei denen die Knoten keinen Konsens finden.

Mit dem vierten Punkt von der vorherigen Seite wird gezeigt, dass Iroha nur Blöcke zu Blockchain hinzufügt, wenn mehr als die Hälfte der Knoten im Netzwerk diesen validieren. Allerdings erfüllt dieses Verhalten nicht die byzantinische Fehlertoleranz nach der Spezifikation von YAC [17] und der allgemeingültigen Theorie dahinter [13]. Mit dem aktuell implementierten System von Iroha kann der byzantinische Fehler nicht ausgeschlossen werden. Somit wird Hypothese 1 falsifiziert.

5.2.2 Hypothese 2

Daten innerhalb eines Blockes sind von den betroffenen Personen autorisiert, sind stets authentisch und konform mit den Vorgaben von DIVA.EXCHANGE.

Mithilfe der ersten Erkenntnis kann gezeigt werden, dass nur der Knoten selbst Zugriff auf seinen privaten Schlüssel und seine Kopie der Blockchain hat. In Verbindung mit der dritten Erkenntnis kann somit belegt werden, dass nach öffentlichem Wissen nur der Ersteller selbst eine gültige Signatur anbringen kann. Auch der Block selbst ist dank dem Hash Algorithmus geschützt, sodass jede nachträgliche Veränderung festzustellen ist. Zudem bietet die Diva-API keine Funktionalität zur Anpassung der schon bestehenden Blockchain. Hypothese 2 kann somit nicht falsifiziert werden.

5.2.3 Hypothese 3

Das Quorum ist resistent gegen böswillige Manipulationen.

Punkt vier und fünf zeigen, dass das Quorum nur $2f+1$ beträgt. Das heisst, dass sobald mehr als die Hälfte der Knoten einen Block verifizieren, wird dieser Block in die Blockchain aufgenommen. Zudem werden die Knoten nicht entfernt, wenn sie für längere Zeit nicht aktiv sind, siehe Punkt 6. Somit wäre das Quorum zwar ungenügend, aber resistent gegen Manipulationen. Der sechste Punkt auf der vorherigen Seite zeigt allerdings auch, dass Knoten aktiv aus dem Netzwerk entfernt werden können. Jeder Knoten kann jeden anderen Knoten entfernen lassen, und die anderen Knoten stimmen dieser Entfernung standardgemäß zu. Ein Angreifer kann dadurch so lange Knoten entfernen, bis er die Mehrheit der Knoten im Netzwerk kontrolliert. Müsste eine Entfernung eines Knoten von ihm selbst ausgehen, wäre das implementierte System sicherer. Zum Beispiel könnte ein Knoten seine eigene Entfernung signieren, somit kann nur er sich selbst entfernen.

Ausserdem kann ein Angreifer auch neue Knoten erstellen, bis er mehr als die Hälfte besitzt. Weil Iroha eine Proof of Stake Blockchain implementiert, werden die Kosten einen solchen Angriff eher tief ausfallen. Siehe dazu "Kosten für 51 % Angriff auf Iroha Blockchain" im Anhang.

Bei Proof of Stake Blockchain mit einer internen Kryptowährung würde ein solcher Angriff schlussendlich viel mehr kosten, weil der Angreifer einen bestimmten Betrag von der Kryptowährung besitzen muss, um einen grösseren Einfluss auf die Abstimmung zu haben. Das macht auch den Angriff selbst deutlich weniger lukrativ, weil er mit einem erfolgreichen Angriff gleichzeitig den Wert der Kryptowährung potenziell verringert. Allerdings existieren bei einer Proof of Stake Blockchain mit interner Kryptowährung andere, lukrativere Schwachstellen und Angriffspunkte, welche berücksichtigt werden müssen [26].

Mit den beschriebenen Argumenten wird belegt, dass weder DIVA.EXCHANGE noch Iroha das Quorum ausreichend beschützt und die Hypothese 3 wird somit falsifiziert.

5.2.4 Hypothese 4

DIVA.EXCHANGE verwendet die eigene und externe APIs in sicherer Art und Weise.

Die komplette Überprüfung und Analyse des Source Codes von DIVA.EXCHANGE und Iroha wäre ein zu grosser Aufwand für diese Arbeit. Die untersuchten Aspekte zeigen auf, dass bei der Entwicklung keine Fehlannahmen getroffen worden sind und die analysierten Funktionen korrekt verwendet werden. Bei der Entwicklung der neuen Blockchain von DIVA.EXCHANGE muss ohnehin nochmals untersucht werden, ob die Verwendung der eigenen und externen API sicherheitsgemäss verläuft. Die Hypothese 4 kann somit weder nicht falsifiziert noch falsifiziert werden.

5.2.5 Hypothese 5

Die Knoten sind vollständig anonym und können nicht von anderen Knoten im Netzwerk manipuliert werden.

Der erste Aspekt des Thread-Modellings aus Kapitel 3 widmet sich diesem Aspekt. Dabei wird festgehalten, dass I2P als vollständig anonym gilt und ausreichende Sicherheit aufweist [35]. Weiter wäre auch bei diesem Aspekt eine komplette Untersuchung des I2P Projekts ein zu grosser Arbeitsaufwand für diese Arbeit. Mit dem öffentlichen Wissen, dass I2P vollständig anonymisiert und keine groben Sicherheitslücken aufweist, kann diese These nicht falsifiziert werden.

5.3 Fazit

Mit der aktuellen Implementation von DIVA.EXCHANGE und dem Hyperledger Iroha sind folgende drei Angriffe nicht auszuschliessen. Die ersten zwei Angriffe hätten gravierende Auswirkungen auf das Vertrauen in DIVA.EXCHANGE und es wird stark empfohlen, die entsprechenden Schwachstellen zu diesen Angriffen zu schliessen. Der dritte Angriff kann je nach Implementation der Diva-API und der von DIVA.EXCHANGE verwendeten Blockchain leichte bis sehr gravierende Auswirkungen haben. Die detaillierten Empfehlungen zur Entfernung der Schwachstellen folgt im nächsten Kapitel.

Erstens kann ein byzantinischer Fehler mit einer einfachen Mehrheit nicht ausgeschlossen werden, also mit dem $2f+1$ Prinzip. Um zu garantieren, dass alle Knoten denselben Stand haben und neue Blöcke nicht fehlerhaft oder von Angreifern absichtlich manipuliert sind, müssen mehr als zwei Drittel der Knoten neuen Blöcken zustimmen, gemäss dem $3f+1$ Prinzip [13].

Zweitens kann ein Angreifer beliebig neue Knoten ins Netzwerk hinzufügen und andere Knoten, die er nicht kontrolliert, entfernen lassen. Auch wenn das Entfernen zukünftig sicherer ist, also dass ein Knoten nur sich selbst entfernen kann, kann ein solcher Angriff erfolgreich sein. Weil es sich um eine Proof of Stake Blockchain handelt, ist das Hinzufügen von genügend Knoten eher kostengünstig. Siehe dazu "Kosten für 51 % Angriff auf Iroha Blockchain" im Anhang.

Drittens könnte ein Angreifer jedes Mal zum Abstimmungsorganisator bestimmt werden, wenn er die kleinste Latenz zur Diva-API hat. Ob diese Schwachstelle in Zukunft existiert, ist davon abhängig, ob bei DIVA.EXCHANGE in der realen Welt irgendein zentraler Service läuft, wie beim gezeigten Testnetz. Wenn die Logik vollständig verteilt ist und jeder Knoten seine eigene Diva-API besitzt, wie im Data-Flow-Diagramm gezeigt wird, könnte der Angreifer nur Abstimmungen organisieren, die von ihm selbst ausgehen. Dies, weil der Iroha Prozess und DIVA.EXCHANGE Prozess bei jedem Knoten lokal in je einem Docker Container laufen, und somit eine viel kleinere Latenz haben, im Vergleich zur Kommunikation über das I2P zwischen Knoten. Auch wenn der Angreifer nun seine Abstimmungen selbst organisieren kann, bräuchte er trotzdem eine $2f+1$ respektive $3f+1$ Mehrheit um beliebige Einträge erstellen, und der entscheidende Aspekt ist somit diese Mehrheit. Trotzdem sollte diese Schwachstelle behoben werden und jede Abstimmung soll von einem zufälligen Knoten geleitet werden, damit im Normalfall der von DIVA.EXCHANGE definierte Abstimmungsablauf durchgeführt wird.

5.4 Empfehlungen an DIVA.EXCHANGE

Um DIVA.EXCHANGE sicherer zu gestalten, sind die sechs Aspekte auf den folgenden Seiten zu berücksichtigen. Aspekt 1 beschreibt mögliche Verteidigungen gegen "zwei Drittel Angriffe", also komplett Netzwerkübernahmen durch den Angreifer und Verteidigungen gegen Liveness Denial Angriffe, womit die Blockchain eingefroren werden kann. In einer Proof of Stake Blockchain ohne interne Währung existieren keine offensichtlichen oder exakt passenden Verteidigungen gegen diese Angriffe. Deshalb werden die Verteidigungen und die Herausforderungen dazu nochmals genauer analysiert. Im Vergleich dazu sind bei den Aspekten 2 bis 6 alle nötigen Erkenntnisse und Schlussfolgerungen schon vorhanden. Dadurch ist Aspekt 1 deutlich ausführlicher als die Aspekte 2 bis 6.

5.4.1 Aspekt 1 – Quorum und Manipulationsschutz

Das Quorum muss von "mehr als die Hälfte" zu "mehr als zwei Drittel" erhöht werden, also von $2f+1$ zu $3f+1$. Weil planmäßig eine eigene Blockchain bei DIVA.EXCHANGE implementiert wird, muss die neue Blockchain dieses Quorum für neue Blöcke unbedingt erfüllen. Zusätzlich soll Logik eingebaut werden, welche das Beitreten von neuen Knoten in bestimmter Weise kontrolliert. Beispiele für eine solche Überprüfung sind folgend aufgelistet:

- a. Eine Variante wäre, dass pro Stunde nur 1 % neue Knoten hinzugefügt werden können, und somit ein Angriff viel länger dauern würde. Als Beispiel mit 1000 Knoten: In einer Stunde können somit nur 10 neue Knoten hinzugefügt werden. Ein offensichtlicher Nachteil davon ist, dass ein Netzwerk eher langsam wachsen kann, auch wenn viele legitime Benutzer beitreten wollen. Zudem kann ein Angreifer damit einen Beitritt für legitime Benutzer erheblich erschweren, wenn die meisten Beitrittsanfragen von ihm selbst stammen.

Für einen Angriff mit einer $3f+1$ Mehrheit muss ein Angreifer das Doppelte der bestehenden Knoten hinzufügen. Also bei 100 Knoten müsste er mehr als 200 Knoten selbst hinzufügen. Die Formel, wie lange er bräuchte, ist damit folgende [45]:

$$n = \frac{\ln(A) - \ln(P)}{\ln(1+i)}$$

Wobei n die Zeitschritte sind, die der Angreifer benötigt, um genügend Knoten zu erhalten. i steht für die maximale Wachstumsrate, also 1 % nach dem Beispiel von oben. P bezeichnet die bestehenden Knoten, 100 nach dem Beispiel von oben. A steht für die kleinste Netzwerkgröße, bei dem der Angreifer erfolgreich wäre, also 301 Knoten gemäß dem Beispiel von oben. Wegen $3f+1$ entspricht A immer $3P+1$. Daraus entsteht die folgende, allgemeingültige Formel:

$$n = \frac{\ln(3 * P - 1) - \ln(P)}{\ln(1+i)} = \frac{\ln(3 + \frac{1}{P})}{\ln(1+i)} \approx \frac{\ln(3)}{\ln(1+i)}$$

Im Durchschnitt braucht ein Angreifer also 110 Zeitschritte, also 110 Stunden oder 4.6 Tage, bei einem maximalen Wachstum von 1 %, um den Angriff durchzuführen, unabhängig der Netzwerkgröße. Allerdings wird angenommen, dass keine anderen Benutzer in dieser Zeit beitreten. Bei einem Wachstum von 0.1 % bräuchte ein Angreifer 46 Tage, also fast das zehnfache an Zeit. Allerdings bedeutet das auch, dass das Netzwerk 693 Stunden oder fast einen Monat benötigt, um sich in seiner Anzahl an Knoten zu verdoppeln.

- b. Eine weitere Variante wäre, dass der Knoten beim Beitritt etwas in der Art eines Captchas lösen muss, oder eine kleine Art von Proof of Work, wie zum Beispiel einen bestimmten Hash erstellen oder die x-te Primzahl berechnen. Weil das aber einmalig ist, behält die Blockchain immer noch den Status einer Proof of Stake Blockchain. Der Nachteil hierbei ist, dass ein Angreifer nur etwas mehr Zeit und Aufwand benötigt, um dem Netzwerk beizutreten. Im Vergleich zur Variante a wird einem Benutzer aber nie der Zutritt wegen eins maximalen Wachstumes verweigert.
- c. Eine etwas andere Vorgabe wäre, dass der Benutzer, der DIVA.EXCHANGE betreten will, ein bestimmtes Betriebssystem oder bestimmte Software haben muss. Somit können zum Beispiel IoT Geräte ausgeschlossen werden, weil diese meist nur für Botnets und nicht für legitime Nutzung verwendet werden. Es wird vermutet, dass nur die wenigsten legitimen Benutzer damit ausgeschlossen würden. Andererseits können solche Betriebssystem- oder Software-Checks mit etwas Aufwand ausgetrickst werden, je nachdem, wie gut der Check implementiert ist.

Mit keiner der Varianten a, b oder c kann ein Sybil Angriff mit Garantie verhindert werden. Also ein Angriff, bei dem der Angreifer Knoten hinzufügt, bis er das Netzwerk komplett kontrolliert. Allerdings können die Kosten für einen erfolgreichen Angriff deutlich erhöht werden, ohne den Beitritt in das DIVA.EXCHANGE Netzwerk für legitime Benutzer merklich zu erschweren. Eine Kombination dieser oder ähnlichen Sicherheitsmaßnahmen soll implementiert werden, um mögliche Sybil Angriffe deutlich zu erschweren.

Mit einem 3f+1 System benötigt ein Angreifer allerdings nur einen Drittel der Knoten, um einen Liveness Denial Angriff auszuführen, also dass das Netzwerk keine neuen Blöcke aufnehmen kann. Eine Art von Checkpoints kann dabei Abhilfe schaffen. Bei anderen Proof of Stake werden diese Checkpoints auch verwendet [26], allerdings gegen andere Angriffe und mit einer leicht anderen Funktionalität als der folgende Entwurf:

In der Diva-API wird jede Minute ein Ping ausgesendet und folgend in die Blockchain aufgenommen. Wird das Netzwerk allerdings mit einem Liveness Denial Angriff lahmgelegt, werden auch die Pings nicht mehr angenommen. Somit kann definiert werden, dass wenn die Pings für eine gewisse Zeit ausfallen, die Blockchain angepasst werden muss. Bei dieser Anpassung können zum Beispiel die letzten x Beiträge aus der Blockchain gelöscht werden, somit soll sichergestellt werden, dass der Angreifer wieder weniger als einen Drittel der Knoten kontrolliert. Einige Blöcke müssen somit neu gehasht werden und die "previous block hash" der folgenden Blöcke muss angepasst werden. Wenn sich nun mehr als zwei Drittel der Knoten legitim verhalten, erstellen diese Knoten alle dieselbe neue Blockchain und können sich auf diese angepasste Blockchain einigen. Sind zu wenige Beiträge, oder die Beiträge von legitimen Benutzern gelöscht worden, sodass der Angreifer immer noch mehr als einen Drittel der Knoten kontrolliert, muss dieser Vorgang wiederholt werden, bis die Blockchain wieder normal funktionieren kann und die Pings wieder eintreffen.

Mit dieser Methode kann nicht garantiert werden, dass die Knoten des Angreifers entfernt werden, es kann nur garantiert werden, dass das Netzwerk irgendwann wieder funktioniert. Einen Vorteil für eine komplette Netzwerkübernahme entsteht dadurch auch nicht, weil die Knoten chronologisch entfernt werden. Das heißt kontrolliert der Angreifer nach einer Entfernung plötzlich mehr als zwei Drittel der Knoten, war das schon einmal der Fall.

Zusätzlich soll vermerkt werden, dass die Knoten, ausser dem absichtlichen Weglassen einer Antwort, wie bei einem Liveness Denial, auch unabsichtlich deaktiviert sein können. Ein Benutzer kann zum Beispiel einen Knoten erstellen und dann den Zugang zum PC verlieren oder der PC funktioniert plötzlich nicht mehr, aber der Knoten ist noch im DIVA.EXCHANGE Netzwerk registriert.

Eine Funktion könnte implementiert werden, dass Knoten einmal pro Woche bis einmal pro Monat einen Ping aussenden müssen, ansonsten werden sie automatisch aus dem Netzwerk ausgeschlossen. Mit dieser erhöhten Zeitspanne ist es praktisch unmöglich, dass ein Knoten gegen seinen Willen entfernt wird. Ein absichtliches und komplettes Blockieren des Pings durch einen Angreifer ist über eine solche Zeitspanne dank dem I2P Netzwerk praktisch unmöglich [25]. Auch unabsichtlich lassen normale Benutzer ihren PC nicht eine Woche oder gar einen Monat ausgeschaltet. Auch wenn dies teilweise vorkommt, wird das voraussichtlich nicht häufig genug sein, um den zwei Dritteln Angriff zu ermöglichen. Zusätzlich muss der Benutzer nur wieder eine Beitrittsanfrage erstellen, um wieder im Netzwerk zu sein.

5.4.2 Aspekt 2 – Auswahl des Abstimmungsorganisators

Der Knoten, der die Abstimmung für einen neuen Knoten organisiert, muss per Zufall ausgewählt werden. Ansonsten ist nicht garantiert, dass auch die abstimgenden Knoten gemäss der Spezifikation von DIVA.EXCHANGE gewählt werden. Es lässt sich nicht verhindern, dass per Zufall ein Knoten eines Angreifers die Abstimmung leiten darf, das System ist damit aber trotzdem sicherer.

5.4.3 Aspekt 3 – Performance bei Abstimmungen

Bei jeder Abstimmung wird eine gewisse Menge an Knoten ausgewählt, die einen Block validieren sollen. Gemäss den Resultaten und der Spezifikation von YAC wird eine Menge an Knoten bestimmt, welche nacheinander abstimmen dürfen. Antwortet ein Knoten dieser Menge nicht innerhalb einer gewissen Zeit, wird eine komplett neue Menge an Knoten bestimmt. Wie in den Resultaten gezeigt steigt somit die Laufzeit exponentiell an im Verhältnis zur Netzwerkgröße und nicht-antwortenden Knoten. Eine viel bessere Funktionsweise wäre, sofort alle Knoten zur Abstimmung zulassen und warten, bis mehr als zwei Dritteln der Stimmen eingetroffen sind, unabhängig der Reihenfolge. Ein kleiner Nachteil ist, dass die Abstimmung von Knoten mit kleineren Latenzen häufiger miteinbezogen werden, im Vergleich zu Knoten mit höheren Latenzen. Allerdings ist das nicht gravierend, weil ein Angreifer sowieso mehr als zwei Dritteln der Knoten für eine komplette Netzwerübernahme bräuchte, und nicht die kleinere Latenzen. Und wenn er das Netzwerk blockieren will, wird er einfach nicht antworten, womit die kleineren Latenzen erneut nichts bringt. Mögliche Verteidigung gegen diese beiden Angriffe sind im Aspekt 1 aufgelistet.

5.4.4 Aspekt 4 – Entfernung von Knoten

Das Entfernen von Blöcken muss sicherer gestaltet werden, anderenfalls ist es schwierig zu garantieren, dass die Benutzer immer Zugriff auf die Blockchain haben und dass keine Manipulationen am Quorum stattfinden können, wie beim Aspekt 1 gezeigt wird. Wenn es gewünscht ist, dass jeder Knoten die Entfernung eines anderen Knoten veranlassen kann, sollten die Knoten im Netzwerk diese Entfernung wenigstens überprüfen. Als Beispiel: Ein Knoten kann nicht entfernt werden, wenn er noch aktiv an Abstimmungen teilnimmt. Oder es wird gesichert, dass ein Knoten nicht beliebig viele andere Knoten nacheinander entfernen kann, zum Beispiel nur einen pro Tag.

5.4.5 Aspekt 5 – Externe APIs und API Abuse

Bei der Verwendung von externen API für die neue Blockchain muss erneut untersucht werden, ob diese nach öffentlichem Wissen als sicher angesehen werden. Zudem darf diese Blockchain keine falschen Annahmen zur Sicherheit der Diva-API treffen. Die Diva-API muss weiterhin in sicherer Art und Weise verwendet und erweitert werden.

5.4.6 Aspekt 6 – DIVA.EXCHANGE Entwickler

Weiterhin soll der Source Code von DIVA.EXCHANGE regelmässig überprüft werden, um absichtlich eingebaute Schwachstellen zu verhindern. Zudem sollen die Entwickler von DIVA.EXCHANGE auf (Spear-) Phishing Attacks und Richtlinien für sichere Passwörter und deren Umgang [36] aufgeklärt werden.

6 Verzeichnisse

6.1 Literaturverzeichnis

- [1] K. Bächler und C. Bächler-Schenk, „DIVA.EXCHANGE: Freie Banking-Technologie. Für alle.“, *DIVA.EXCHANGE*. <https://www.diva.exchange/de/> (zugegriffen Mai 21, 2021).
- [2] „Hyperledger Iroha“, *Hyperledger*. <https://www.hyperledger.org/use/iroha> (zugegriffen Apr. 08, 2021).
- [3] „Common Vulnerabilities and Exposures - Bitcoin Wiki“. https://en.bitcoin.it/wiki/Common_Vulnerabilities_and_Exposures#CVE-2010-5139 (zugegriffen Mai 20, 2021).
- [4] „What Happens To Lost Bitcoins?“, *ReadWrite*, Jan. 13, 2014. <https://readwrite.com/2014/01/13/what-happens-to-lost-bitcoins/> (zugegriffen Mai 20, 2021).
- [5] S. Marić, „Untersuchung eines vollständig dezentralen, nicht-diskriminierenden und Privatsphäre-schützenden Handelsnetzwerk für digitale Werte („Krypto-Anlagen“)“, *Codeberg.org*, Jan. 07, 2020. <https://codeberg.org/diva.exchange/academia> (zugegriffen Feb. 25, 2021).
- [6] „Einführung in DIVA.EXCHANGE: LIGHTNING TALK FOSDEM 2021“, *DIVA.EXCHANGE*. <https://www.diva.exchange/de/privatsphaere/eine-einfuehrung-in-diva-exchange-transcript-des-lightning-talks-an-der/> (zugegriffen Mai 31, 2021).
- [7] W. Kenton, „What is Hyperledger Iroha?“, *Investopedia*. <https://www.investopedia.com/terms/h/hyperledger-iroha.asp> (zugegriffen Apr. 08, 2021).
- [8] N. Iushkevich, A. Lebedev, R. Šketa, und M. Takemiya, *D3ledger: The Decentralized Digital Depository Platform for Asset Management Based on Hyperledger Iroha*. 2019. doi: 10.18690/978-961-286-282-4.4.
- [9] „Distributed Ledger: Die Technologie hinter den virtuellen Währungen am Beispiel der Block-chain“, *BaFin*. https://www.bafin.de/SharedDocs/Veroeffentlichungen/DE/Fachartikel/2016/fa_bj_1602_blockchain.html (zugegriffen Mai 30, 2021).
- [10] „ITU-T Recommendation database“, *ITU*. <https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=14332&lang=en> (zugegriffen Mai 30, 2021).
- [11] L. Lamport, R. Shostak, und M. Pease, „The Byzantine Generals Problem“, *ACM Transactions on Programming Languages and Systems*, Bd. 4, Nr. 3, S. 20.
- [12] L. M. Bach, B. Mihaljevic, und M. Zagar, „Comparative analysis of blockchain consensus algorithms“, in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Mai 2018, S. 1545–1550. doi: 10.23919/MIPRO.2018.8400278.
- [13] P. Feldman und S. Micali, „An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement“, *SIAM J. Comput.*, Bd. 26, Nr. 4, S. 873–933, Aug. 1997, doi: 10.1137/S0097539790187084.
- [14] „Consensus Algorithms in Blockchain“, *GeeksforGeeks*, Apr. 25, 2019. <https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/> (zugegriffen Apr. 26, 2021).
- [15] „proof-of-stake-faqs“, *Ethereum Wiki*. <https://eth.wiki/en/concepts/proof-of-stake-faqs> (zugegriffen Mai 13, 2021).
- [16] „Consensus Algorithms: Proof-of-Stake & Cryptoeconomics“, *Nichanan Kesopat*. <https://www.nichanank.com/blog/2018/6/4/consensus-algorithms-pos-dpos> (zugegriffen Mai 13, 2021).
- [17] F. Muratov, A. Lebedev, N. Iushkevich, B. Nasrulin, und M. Takemiya, „YAC: BFT Consensus Algorithm for Blockchain“, *arXiv:1809.00554 [cs]*, Sep. 2018, Zugegriffen: März 18, 2021. [Online]. Verfügbar unter: <http://arxiv.org/abs/1809.00554>
- [18] V. Brühl, „Bitcoins, Blockchain und Distributed Ledgers: Funktionsweise, Marktentwicklungen und Zukunftsperspektiven“, *Wirtschaftsdienst*, Bd. 97, Nr. 2, S. 135–142, Feb. 2017, doi: 10.1007/s10273-017-2096-3.
- [19] „Are blockchains safe? How to attack and prevent attacks“. <https://www.seba.swiss/research/are-blockchains-safe-how-to-attack-them-and-prevent-attacks> (zugegriffen März 31, 2021).

- [20] S. Konst, „Sichere Log-Dateien auf Grundlage kryptographisch verketteter Einträge“, S. 81.
- [21] N. Popper, „Decoding the Enigma of Satoshi Nakamoto and the Birth of Bitcoin“, *The New York Times*, Mai 15, 2015. Zugriffen: März 27, 2021. [Online]. Verfügbar unter: <https://www.nytimes.com/2015/05/17/business/decoding-the-enigma-of-satoshi-nakamoto-and-the-birth-of-bitcoin.html>
- [22] S. Nakamoto, „Bitcoin: A Peer-to-Peer Electronic Cash System“, S. 9.
- [23] M. Rennhard und S. Neuhaus, „Secure-Development-Lifecycle“. ZHAW / SoE / InIT, Aug. 01, 2020.
- [24] „What is a DDoS attack?“ <https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by-norton.html> (zugriffen März 31, 2021).
- [25] „The Network Database - I2P“. <https://geti2p.net/en/docs/how/network-database#threat> (zugriffen Mai 26, 2021).
- [26] E. Deirmentzoglou, „Rewriting History: A Brief Introduction to Long Range Attacks“, *Medium*, Mai 31, 2018. <https://blog.positive.com/rewriting-history-a-brief-introduction-to-long-range-attacks-54e473acdba9> (zugriffen Mai 27, 2021).
- [27] „Sybil attack“, *Wikipedia*. Mai 23, 2021. Zugriffen: Mai 28, 2021. [Online]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=Sybil_attack&oldid=1024617448
- [28] „Birthday attack“, *Wikipedia*. Mai 19, 2021. Zugriffen: Mai 28, 2021. [Online]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=Birthday_attack&oldid=1023994495
- [29] „Collision attack“, *Wikipedia*. Apr. 25, 2021. Zugriffen: Mai 28, 2021. [Online]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=Collision_attack&oldid=1019813321
- [30] „Announcing the first SHA1 collision“, *Google Online Security Blog*. <https://security.google-blog.com/2017/02/announcing-first-sha1-collision.html> (zugriffen Apr. 01, 2021).
- [31] „CERT/CC Vulnerability Note VU#836068“. <https://www.kb.cert.org> (zugriffen Apr. 01, 2021).
- [32] „Replay attack“, *Wikipedia*. Mai 10, 2021. Zugriffen: Mai 28, 2021. [Online]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=Replay_attack&oldid=1022505726
- [33] „Keylength - BSI Cryptographic Key Length Report (2020)“. <https://www.keylength.com/en/8/> (zugriffen Apr. 01, 2021).
- [34] „Software supply chain attacks – everything you need to know“, *The Daily Swig / Cybersecurity news and views*, Feb. 11, 2021. <https://portswigger.net/daily-swig/software-supply-chain-attacks-everything-you-need-to-know> (zugriffen Apr. 15, 2021).
- [35] „I2P“, *Wikipedia*. Apr. 08, 2021. Zugriffen: Apr. 14, 2021. [Online]. Verfügbar unter: <https://en.wikipedia.org/w/index.php?title=I2P&oldid=1016635535>
- [36] „Basic security practices regarding passwords and online identities“ <https://www.enisa.europa.eu/media/news-items/basic-security-practices-regarding-passwords-and-online-identities> (zugriffen Apr. 26, 2021).
- [37] diva.exchange, „diva-dockerized“, *Codeberg.org*. <https://codeberg.org/diva.exchange/diva-dockerized> (zugriffen März 29, 2021).
- [38] „EdDSA“, *Wikipedia*. Mai 07, 2021. Zugriffen: Mai 17, 2021. [Online]. Verfügbar unter: <https://en.wikipedia.org/w/index.php?title=EdDSA&oldid=1021908408>
- [39] „ed25519.js vulnerabilities“, *Snyk*. <https://snyk.io/vuln/npm:ed25519.js> (zugriffen Mai 17, 2021).
- [40] „SHA-3“, *Wikipedia*. Mai 17, 2021. Zugriffen: Mai 17, 2021. [Online]. Verfügbar unter: <https://en.wikipedia.org/w/index.php?title=SHA-3&oldid=1023600862>
- [41] „js-sha3 vulnerabilities“, *Snyk*. <https://snyk.io/vuln/npm:js-sha3> (zugriffen Mai 17, 2021).
- [42] Stand-up Maths, *How lucky is too lucky?: The Minecraft Speedrunning Dream Controversy Explained*, (Feb. 04, 2021). Zugriffen: Mai 31, 2021. [Online Video]. Verfügbar unter: <https://youtu.be/8Ko3TdPy0TU?t=1799>
- [43] M. Rennhard und S. Neuhaus, „Security-Design-Principles“. ZHAW / SoE / InIT, Aug. 01, 2020.
- [44] D. Goodin, „Hackers backdoor PHP source code after breaching internal git server“, *Ars Technica*, März 29, 2021. <https://arstechnica.com/gadgets/2021/03/hackers-backdoor-php-source-code-after-breaching-internal-git-server/> (zugriffen Mai 17, 2021).

- [45] „Calculating the period of an investment | Finance | Siyavula“. <https://intl.siyavula.com/read/math/grade-12/finance/03-finance-01> (zugegriffen Mai 29, 2021).
- [46] J. Porras, J. Päkäläinen, J. Khakurel, und A. Knutas, „Security In The Internet Of Things – A Systematic Mapping Study“, S. 10.
- [47] X. Zhang, O. Upton, N. L. Beebe, und K.-K. R. Choo, „IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers“, *Forensic Science International: Digital Investigation*, Bd. 32, S. 300926, Apr. 2020, doi: 10.1016/j.fsidi.2020.300926.
- [48] „High Memory Instances“, Linode. <https://www.linode.com/products/high-memory/> (zugegriffen Mai 27, 2021).
- [49] „How Much Does A Botnet Cost?“ <https://threatpost.com/how-much-does-botnet-cost-022813/77573/> (zugegriffen Mai 27, 2021).

6.2 Glossar

Begriff	Erklärung
DIVA.EXCHANGE	Ein vollständig verteiltes, Blockchain-basiertes, Open Source Handelsnetzwerk. Siehe 2.1.1
Hyperledger Iroha	Eine eigenständige Implementation eines vollständig verteilten Ledgers mit Identitätsmanagement durch die Blockchain Technologie. Siehe 2.1.2
Open Source	Als Open Source wird Software bezeichnet, deren Quelltext öffentlich und von Dritten eingesehen, geändert und genutzt werden kann.
Block	Ein Block meint einen Teil einer Blockchain. Eine Datenstruktur bestehend aus hauptsächlich Transaktionen und Hashes.
Blockchain	Eine Blockchain ist eine kontinuierlich erweiterbare Liste von Datensätzen, Blöcke, die mittels kryptographischer Verfahren miteinander verkettet sind. Siehe 2.2.3
I2P	I2P ist ein anonymes, pseudonymes und dezentrales ohne Server auskommendes Rechnernetz, basierend auf freier Software.
Wallet	Eine Anwendung, die als virtuelle Depot für digitale Wertmittel dient. Meist auch Benutzeroberfläche für ein Shared Ledger.
Knoten	Ein Teilnehmer eines Netzwerkes.
Arithmetischer Überlauf	Ein arithmetischer Überlauf tritt auf, wenn das Ergebnis einer Berechnung für den gültigen Zahlenbereich zu groß ist, um noch richtig interpretiert werden zu können.
Replay Angriff	Eine Angriffsform auf die Authentizität von Daten in einem Kommunikationsprotokoll. Hierbei sendet der Angreifer zuvor aufgezeichnete Daten, um etwa eine fremde Identität vorzutäuschen.
Interne oder native Kryptowährung	Hat eine Blockchain interne Währung, kann meist nur mit dieser gehandelt werden, im Vergleich zu DIVA.EXCHANGE, welche keine interne Währung aufweist.
Validator	Ein Knoten, der die Aufgabe übernimmt, Transaktionen nach ihrer Richtigkeit zu prüfen.
Quorum	Eine Komponente zur Wahrung der Datenintegrität im Fall eines Teilausfalls, zu dessen Zeitdauer weiterhin Daten geschrieben werden könnten.
API	Eine Programmierschnittstelle. Auch Anwendungsschnittstelle. Genauer: Schnittstelle zur Programmierung von Anwendungen.
Shared Ledger	Beschreibt eine Technik, die für die Dokumentation bestimmter Transaktionen benutzt wird. Im Gegensatz zum klassischen Ansatz, bei dem ein Haupt-

	buch in der Regel von nur einer Instanz verwaltet wird, werden hier dezentral beliebig viele, prinzipiell gleichgestellte Kopien des Ledgers von unterschiedlichen Parteien unterhalten. Siehe 2.2.1
<i>Coin/Tokens</i>	Eine Einheit einer Kryptowährung.
<i>Query</i>	Eine spezifisch gestellte Abfrage an eine Datenbank
<i>Konsensverfahren / Konsensalgorithmus</i>	Ein Mechanismus, der es Knoten ermöglicht, sich in einer verteilten Umgebung zu koordinieren. Er stellt sicher, dass sich alle Knoten im System auf eine Version einer Blockchain einigen. Siehe 2.3.2
<i>Byzantinische Fehlertoleranz</i>	Die maximale Anzahl an unehrlichen Nutzern, die ein verteiltes System verträgt, bevor Fehler nicht mehr ausgeschlossen werden können. Unehrliche Teilnehmer muss kleiner sein als 1/3 aller Nutzer. Siehe 2.3.1
<i>Proof of Stake</i>	Eine Art Konsensalgorithmen, die auf gewichtetem Zufall basieren. Siehe 2.3.2
<i>Proof of Work</i>	Eine Art Konsensalgorithmen, die auf Rechenleistung basieren.
<i>YAC</i>	"Yet Another Distributed Consensus Algorithm" und ist ein Proof of Stake verfahren und wird von Hyperledger Iroha verwendet. Siehe 2.3.3
<i>Proposal</i>	Bei YAC ein vorgeschlagener Block über dessen Freigabe abgestimmt wird.
<i>Hash & Hash Collision</i>	Eine Funktion, die Eingabewerte zu scheinbar zufälligen Anordnungen streut. Bei neuen Versionen wie SHA2 und SHA3 sind noch nie zwei unterschiedliche Eingabewerte gefunden worden, die einen identischen Hashwert ergeben, also eine Hash Collision [40]. Siehe 2.6.3
<i>Commit Nachricht</i>	Eine Nachricht, die die bestätigende Freischaltung einer oder mehrerer Änderungen beschreibt
<i>Schlüsselpaar</i>	Ein paar aus Private Key und Public Key. Dient für ein kryptographisches Verfahren, bei dem die kommunizierenden Parteien keinen gemeinsamen geheimen Schlüssel zu kennen brauchen. Jeder Benutzer erzeugt sein eigenes Schlüsselpaar, das aus einem geheimen Teil und einem nicht geheimen Teil besteht, um zu kommunizieren.
<i>Header</i>	Die Bezeichnung Header steht für Zusatzinformationen (Metadaten), die Nutzdaten am Anfang eines Datenblocks ergänzen.
<i>API Abuse</i>	Fehlerhafte Annahme zur Funktionsweise einer API. Siehe 2.6.1
<i>(D)DoS Attack</i>	Im Fall einer durch eine Vielzahl von gezielten Anfragen verursachten, mutwilligen Dienstblockade spricht man von einer (Distributed)-Denial-of-Service-Attacke. Siehe 2.6.2
<i>Signature Forging</i>	Signature Forging beschreibt einen Angriff, bei dem versucht wird, die Signatur eines anderen Benutzers nachzustellen. Siehe 2.6.5
<i>Supply Chain Attack</i>	Ist ein Angriff, der die verwendete Software anzugreifen, und nicht das Produkt selbst. Siehe 2.6.6
<i>Abuse-Case</i>	Ist ein Spezifikationsmodell für Sicherheitsanforderungen. Ein Abuse-Case beschreibt, wie Software angegriffen und missbraucht werden könnte. So mit können Sicherheitslücken aufgezeigt werden. Siehe 3.1.2

6.3 Abbildungsverzeichnis

Abbildung 1: DIVA.EXCHANGE [6].....	11
Abbildung 2: Sendevorgang [18].....	17
Abbildung 3: Blocks [18]	18
Abbildung 4: Angreifer und Ziel erstellen einen Block.....	22
Abbildung 5: Angreifer löscht den Block.....	22
Abbildung 6: Neue, valide Blockchain.....	22
Abbildung 7: Abuse-Case Diagramm.....	26
Abbildung 8: Data-Flow-Diagram.....	28
Abbildung 9: Angriff via I2P.....	29
Abbildung 10: Supply Chain Attack via Docker Hub oder Codeberg	30
Abbildung 11: Angriff auf lokale Blockchain	30
Abbildung 12: Angriff auf lokale Daten.....	31
Abbildung 13: Anderen Benutzer imitieren.....	31
Abbildung 14: Neue Einträge blockieren	32
Abbildung 15: Legitime Knoten ausschliessen.....	33
Abbildung 16: Angriff mithilfe gewisser Mehrheit	34
Abbildung 17: Teststand, Aufbau und Ablauf	35
Abbildung 18: Neue Einträge blockieren	37
Abbildung 19: Legitime Knoten ausschliessen.....	39
Abbildung 20: Legitime Knoten aktiv entfernen	40
Abbildung 21: Angriff mithilfe gewisser Mehrheit	41
Abbildung 22: Ping Call Stack (1/2)	44
Abbildung 23: Ping Call Stack (2/2)	45
Abbildung 24: T1 - Ping vs. Kein Ping	48
Abbildung 25: T1 - Signaturen.....	49
Abbildung 26: T1 - Performance	50
Abbildung 27: T2 - Entfernt vs. Nicht Entfernt.....	52
Abbildung 28: T2 - Signaturen.....	53
Abbildung 29: T2 - Performance	54
Abbildung 30: T3 - Ping vs. Kein Ping	55
Abbildung 31: T3 - Signaturen.....	56
Abbildung 32: T3 - Performance	57
Abbildung 33: Klassendiagramm der Tests.....	83

7 Anhang

7.1 Projektmanagement

Während der Dauer des Projekts hat sich das Projektteam wöchentlich mit den Betreuern getroffen. An diesen Besprechungen geben die Betreuer Feedback zum Stand des Projekts, offene Aspekte der Arbeit werden abgeklärt und neue Erkenntnisse zum Projekt werden präsentieren.

7.1.1 Aufgabenstellung

The screenshot shows a web-based administration interface for a Bachelor's thesis. At the top, there are logos for Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) School of Engineering and DEPT. T ADMIN TOOLS. The main title is 'Bachelorarbeit 2021 - FS: BA21_neut_02'. Below the title, there are several sections with input fields:

- Allgemeines:**
 - Titel:** Teststand für ein dezentrales und Blockchain-basiertes Handelsnetzwerk für digitale Werte
 - Anzahl Studierende:** 2
 - Durchführung in Englisch möglich:** Ja, die Arbeit kann vollständig in Englisch durchgeführt werden und ist auch für Incomings geeignet.
- Betreuer:**
 - HauptbetreuerIn:** Stephan Neuhaus, neut
 - Studenten-Vereinbarungen:** für diese Arbeit wurde noch keine Vereinbarung getroffen!
- Fachgebiet:**
 - IS Information Security
 - Studiengänge:** IT Informatik
- Zuordnung der Arbeit :**
 - InIT Institut für angewandte Informationstechnologie
 - Infrastruktur:** benötigt keinen zugeteilten Arbeitsplatz an der ZHAW
- Interne Partner :**
 - Es wurde kein interner Partner definiert!
 - Industriepartner:** Es wurden keine Industriepartner definiert!
- Beschreibung:**

DIVA.EXCHANGE (<https://diva.exchange>) entwickelt ein vollständig dezentrales Handelsnetzwerk für digitale Werte. Technisch besteht diese freie und quelloffene Software aus einer Anonymisierungsschicht, einer auf einer Blockchain basierenden Datenhaltung und der darauf aufbauenden Handels- und Verwaltungssoftware.

Die Software ist komplex. Darum wird ein Teststand benötigt. Dieser Teststand soll es ermöglichen, das System bestimmten automatisierten Angriffen auszusetzen. Die Fragen, die dabei untersucht werden sollen, sind u.a.:

 - * Wie viele unehrliche Teilnehmer verträgt das System tatsächlich?
 - * Sind die verwendeten Softwarekomponenten geeignet, die hohen Sicherheitsanforderungen, die an das System gestellt werden, zu erfüllen?
 - * Wie gross ist der Geschwindigkeitszuwachs, wenn mehr Netzwerkknoten zur Verfügung stehen?

Zu einigen dieser Fragen gibt es bereits theoretische Untersuchungen.

Die Ziele der Arbeit sind:

 - * Verständnis des Produktes und der grundlegenden Protokolle
 - * Abstraktion und Formalisierung von Angriffsszenarien
 - * Erstellen eines Teststands mit definierbaren (d.h. konfigurierbaren) Angriffs-Szenarien und definierten Testerwartungen
 - * Ausführen des Teststands mit den definierten Angriffs-Szenarien, beispielsweise ein Replay-Angriff
 - * Zusammenfassung, Dokumentation und Visualisierung der Test-Resultate

Der Teststand selbst soll folgende Bedingungen erfüllen:

 - skalierbares, virtualisiertes Netzwerk mit allen benötigten Knoten/Komponenten - standardmäßig laufen die Knoten in Docker Containern mit dem Original-Quellcode - für die Umsetzung von Angriffs-Szenarien kann der Original-Quellcode abgeändert werden.

Das gesamte DIVA Projekt ist öffentlich, der Quellcode ist hier: <https://codeberg.org/diva.exchange>

Mit den freundlichen Entwicklern kann man jederzeit Kontakt aufnehmen: * Telegram, https://t.me/diva_exchange_chat_de * Mastodon/Fediverse, <https://social.diva.exchange/@social>

At the bottom, there are 'Zurück' and 'Logout' buttons.

7.1.2 Zeitplan

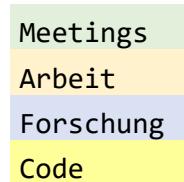
Folgend ist der Zeitplan abgebildet. Dieser ist in der ersten Projektwoche erstellt worden und alle relevanten Termine sind eingehalten worden.

KW	Datum	Tag	Was	Wer
50	16.12.2020	Di	Meat-up with DIVA.EXCHANGE	S,D,B
6	09.02.2021	Di	Organisational Meeting	S,D
8	22.02.2021	Mo	Vorbereitung Kickoff Meeting	S
	22.02.2021	Mo	Kickoff Meeting	S,D,B
	25.02.2021	Do	wöchentliches Meeting	S,D,B
9	-	-	Informationen sammeln zu DIVA.EXCHANGE, Iroha, I2P	S
	01.03.2021	Mo	wöchentliches Meeting	S,D
	04.03.2021	Do	wöchentliches Meeting	S,D,B
10	-	-	Informationen sammeln zu DIVA.EXCHANGE, Iroha, I2P	S
	-	-	BA anfangen zu schreiben, Inhaltsverzeichnis und Struktur	S
	08.03.2021	Mo	wöchentliches Meeting	S,D
	11.03.2021	Do	wöchentliches Meeting	S,D,B
11	-	-	Informationen sammeln zu DIVA.EXCHANGE, Iroha, I2P	S
	-	-	Ausgangslage und Ziel	S
	15.03.2021	Mo	wöchentliches Meeting	S,D
	18.03.2021	Do	wöchentliches Meeting	S,D,B
12	-	-	Use-Cases erstellen	S
	-	-	Theoretische Grundlagen	S
	22.03.2021	Mo	wöchentliches Meeting	S,D
	25.03.2021	Do	wöchentliches Meeting	S,D,B
13	-	-	DFD und Threat-Modelling	S
	-	-	Theoretische Grundlagen	S
	29.03.2021	Mo	wöchentliches Meeting	S,D
	01.04.2021	Do	wöchentliches Meeting	S,D,B
14	-	-	DFD und Threat-Modelling	S
	-	-	Überprüfen und verbessern	S
	05.04.2021	Mo	wöchentliches Meeting	S,D
	08.04.2021	Do	wöchentliches Meeting	S,D,B
15	-	-	Experiment implementieren	S
	-	-	Methodik	S
	12.04.2021	Mo	wöchentliches Meeting	S,D
	15.04.2021	Do	wöchentliches Meeting	S,D,N
16	-	-	Experiment implementieren	S
	-	-	Methodik	S
	19.04.2021	Mo	wöchentliches Meeting	S,D
	22.04.2021	Do	wöchentliches Meeting	S,D,B

	-	-	Experiment implementieren	S
	-	-	Methodik	S
17	26.04.2021	Mo	wöchentliches Meeting	S,D
	29.04.2021	Do	wöchentliches Meeting	S,D,B
	-	-	Experiment durchführen	S
18			Methodik	S
	03.05.2021	Mo	wöchentliches Meeting	S,D
	06.05.2021	Do	wöchentliches Meeting	S,D,B
	-	-	Resultate analysieren	S
19			Resultate	S
	10.05.2021	Mo	wöchentliches Meeting	S,D
	13.05.2021	Do	wöchentliches Meeting	S,D,B
	-	-	Resultate analysieren	S
20			Diskussion	S
	-	-	Arbeit für Feedback einreichen	S,B
	17.05.2021	Mo	wöchentliches Meeting	S,D
	20.05.2021	Do	wöchentliches Meeting	S,D,B
	-	-	Code Dokumentieren	S
21			Überprüfen und verbessern	S
	24.05.2021	Mo	wöchentliches Meeting	S,D
	27.05.2021	Do	wöchentliches Meeting	S,D,B
	-	-	Code Dokumentieren	S
22			Reserve	
	31.05.2021	Mo	wöchentliches Meeting	S,D
	03.06.2021	Do	wöchentliches Meeting	S,D,B
23			Reserve	
	11.06.2021	Fr	Abgabe BA - 17:00 Uhr	S
26	02.07.2021	Do	Präsentation & Verteidigung BA	S

Legende

- S: Studenten
 D: DIVA.EXCHANGE Team
 B: Betreuer, S. Neuhaus



7.1.3 Meeting Notizen

Für das Kickoff Meeting und die erste Besprechung sind ausführliche Notizen erstellt worden. Für die restlichen Besprechungen sind die Notizen in die Protokolle oder direkt in die Arbeit eingeflossen. Die Notizen sind auf den folgenden Seiten ersichtlich.

7.1.3.1 KW 50 – Erstes Meeting mit DIVA.EXCHANGE

Open Source DIVA.EXCHANGE -> sonst niemand angewiesen auf Spenden
oft Zusammenarbeit mit Hochschulen

Aufgabenstellung:

- Teststand um Angriffe simulieren -> Vertrauen erhöhen, keinen Formalen Beweis zur Sicherheit
- praktischer Teil: Vertrauenserhöhung durch abgewiesene Angriffe

DIVA:

- Transparenz, möchte Teststand, App im Moment noch in der Alpha-Phase
- fokussieren auf Netzwerkprobleme und Sicherheit und kreative Sicherheitsfragen angehen

Ablauf:

- Explorationsphase (DIVA und Testnetzwerke verstehen -> <https://testnet.DIVA.EXCHANGE/>, https://drive.google.com/file/d/1JUZei_XUEc3b6hJ83srQEw6yUeXbplaz/view)
- Thread Model erstellen (DFD) und mit STRIDE Bedrohungen finden
- vertikaler Use Case (einfaches Netzwerk aufsetzen und austesten) danach Netzwerk erweitern (skalieren)
- schlussendlich aufzeigen was gemacht worden ist, was man noch machen könnte

7.1.3.2 KW 6 – Kickoff Meeting

Stefan Neuhaus' Tipps

- wöchentliche Meetings
- es gibt keine dummen Fragen
- gute Planung -> Pakete definieren

Todo

- Meetings mit Stefan Neuhaus definieren

Tools

- via git auf Codeberg ein Repo unterhalten
- Arbeit (mit Word) auf Teams unterhalten

7.1.4 Protokolle

Die relevanten Besprechungen mit dem Betreuer oder dem DIVA.EXCHANGE Team sind protokolliert. Auf den Protokollen sind die Pendenzen bis zur nächsten Besprechung festgehalten. Im ersten Absatz sind die Pendenzen aufgelistet, die alle betreffen, und im folgenden Absatz die Pendenzen für das Projektteam. "Kein Protokoll" bedeutet entweder, dass kein Meeting stattgefunden hat oder keine nennenswerten Aspekte besprochen worden sind.

7.1.4.1 KW 8 – 1. Besprechung

Alle

- Meetings wurden festgelegt. Montag 9:30 - 10:00 mit Carolyn und Konrad | Donnerstag 9:30 - 10:00 mit Carolyn, Konrad und Stephan.
- Es wurde entschieden, dass Sascha die Meetings in Zukunft technisch organisiert und 30 Minuten vor dem Meeting den Link dazu per E-Mail versendet.
- Es wurde der Fokus auf den storage-layer besprochen, da Angriffe auf lokale Applikation wenig Sinn machen.
- Der Termin für die Präsentation wurde besprochen. Da dieser aber stark von der ZHAW abhängig ist, wurde beschlossen abzuwarten. Levi und Sascha sollen in der Zwischenzeit aufmerksam beobachten, ob die Hochschulleitung die Studenten genauer informiert.
- Die Bewertung der BA wurde besprochen. Betreuer, DIVA.EXCHANGE und Experte der ZHAW werden dabei teilhaben.
- Der Zeitplan wurde in einem ersten Schritt vorgestellt und bestätigt.
- Ein Protokoll mit Schwerpunkt auf gemeinsame Abmachungen und Entscheidungen wurde angekündigt.

Studenten

- Repo aufsetzen.
- Einlesen.
- Fragen über Telegram oder Mail stellen.
- Plan für den Aufbau des Dokuments zur weiteren Besprechung montags aussuchen.

7.1.4.2 KW 8 – 2. Besprechung

Alle

- Meetings wurden festgelegt. Montag 9:30 - 10:00 mit Carolyn und Konrad | Donnerstag 9:30 - 10:00 mit Carolyn, Konrad und Stephan.
- Die Quellen der Arbeit werden auf einer Liste im Repository für alle festgehalten.
- Es wurde ermöglicht ein Repo auf Codeberg unter DIVA.EXCHANGE zu erstellen.
- Der Plan für den Aufbau des Dokuments wird Stephan per Mail zugesendet und eventuell Donnerstag kurz zusammen angesehen.

Studenten

- Repo aufsetzen und erste Recherche
- Fragen über Telegram oder Mail stellen
- Aufbau des Dokuments zur Überprüfung versenden

7.1.4.3 KW 9 – 3. Besprechung

Kein Protokoll

7.1.4.4 KW 9 – 4. Besprechung

Alle

- vorgeschlagene Struktur der BA ist ok.

7.1.4.5 KW 10 – 5. Besprechung

Kein Protokoll

7.1.4.6 KW 10 – 6. Besprechung

Alle

- Fragen zur API wurden beantwortet. Der Dev-Branch von DIVA.EXCHANGE soll verwendet werden.
- Probleme mit der anzusteuernden IP für das Testnetzwerk wurden besprochen -> normalerweise ist die API unter <http://172.29.101.100:3920/> zu erreichen.
- Byzantinische Fehlertoleranz wurde besprochen, da sie in Zusammenhang mit Hyperledger Iroha eine Schwachstelle offenbaren könnte.

Studenten

- Nächste Woche beginnt der Schreibprozesses der Einleitung

7.1.4.7 KW 11 – 7. Besprechung

Alle

- Es wurde nochmals über die Abgabe gesprochen. Diese erfolgt am 11.6.2021. Ein Datum für die Präsentation ist noch nicht bekannt. Das Zeitfenster beinhaltet die Kalenderwochen 26 und 27.

Studenten

- Nächste Woche beginnt der Schreibprozesses der Einleitung

7.1.4.8 KW 11 – 8. Besprechung

Kein Protokoll

7.1.4.9 KW 12 – 9. Besprechung

Alle

- Es wurde über das Threat-Modeling gesprochen und Stephan hat Empfehlungen betreffend der tiefe gegeben. Dazu kann nicht alles untersucht werden, die relevanten Aspekte müssen zum Teil mit Bauchgefühl ausgewählt werden.
- Angriffe müssen systematisch erfolgen.

Studenten

- Nächste Woche beginnt der Schreibprozesses der Einleitung

7.1.4.10 KW 12 – 10. Besprechung

Alle

- Es wurden einige Fragen zur API und zu den Containern gestellt. Allgemein war die Angriffsfläche Docker Container Gesprächsthema.

Studenten

- Aktuell wird das Kapitel "theoretische Grundlagen" geschrieben. Zusätzlich sollen Use-Cases erstellt werden.

7.1.4.11 KW 13 – 11. Besprechung

Alle

- Die erste Version des Data-Flow-Diagram wurde gezeigt und es wurden zusammen Eckpunkte dieses Diagramms besprochen. "Trust boundaries" einfügen.
- Die Änderungen wurden direkt in das Diagramm übernommen. Zum einen ging es dabei um den transportlayer von Diva, zum anderen um die geschützten Zonen

Studenten

- Aktuell wird das Kapitel "theoretische Grundlagen" geschrieben. Zusätzlich sollen Use-Cases erstellt werden.

7.1.4.12 KW 13 – 12. Besprechung

Alle

- Konrad informierte darüber, dass es bald eine neue Iroha Version geben würde, die uns allerdings nicht beeinflussen sollte.
- Ein erstes Skript zum Aufbau der Tests wurde gezeigt und erläutert.
- Konrad stellt nächste Woche Container zur Verfügung

Studenten

- Aktuell wird das Kapitel "theoretische Grundlagen" geschrieben. Zusätzlich sollen Use-Cases erstellt werden.

7.1.4.13 KW 14 – 13. Besprechung

Kein Protokoll

7.1.4.14 KW 14 – 14. Besprechung

Kein Protokoll

7.1.4.15 KW 15 – 15. Besprechung

Alle

- Das Feedback ist mit Carolyn und Konrad angeschaut worden, wobei es aber keine weiteren Punkte gab ausser die schriftlichen.
- Der Technische Teil der Arbeit wird den Grossteil der Arbeit annehmen, es ist kein Kapitel zu Privatsphäre oder moralischer Diskussion geplant.
- Kapitel über byzantinische Fehlertoleranz und Konsens Algorithmen sind wichtig für die Theorie.

Studenten

- Aktuell werden erste Tests implementiert und das Kapitel Methodik geschrieben.

7.1.4.16 KW 15 – 16. Besprechung

Alle

- Das Feedback ist mit nochmals mit Stephan angeschaut worden, wobei es aber keine weiteren Punkte gab ausser die schriftlichen.
- Kapitel über byzantinische Fehlertoleranz und Konsens Algorithmen sind wichtig für die Theorie.
- Implementation von ersten Tests so fertig wie möglich vorführen beim nächsten Mal.

Studenten

- Aktuell werden erste Tests implementiert und das Kapitel Methodik geschrieben.

7.1.4.17 KW 16 – 17. Besprechung

Alle

- Erste Testumgebung wurde gezeigt.
- Eine Abwesenheit von Carolyn und Konrad in Woche 17 ist besprochen worden.

Studenten

- Aktuell werden erste Tests implementiert und das Kapitel Methodik geschrieben.

7.1.4.18 KW 16 – 18. Besprechung

Alle

- Wir haben besprochen, dass 16 Knoten als zum Beweis reichen. Die Skalierung nach oben ist von dort an sekundär.
- Fokus ist Iroha. Die Diva-API wird erstmal nicht benötigt.
- Das Meeting am Montag 26.4 fällt aus.
- Am 29.4 sollten Experimente fertig sein.

Studenten

- Aktuell werden erste Tests implementiert und das Kapitel Methodik geschrieben

7.1.4.19 KW 17 – 19. Besprechung

Kein Protokoll

7.1.4.20 KW 17 – 20. Besprechung

Alle

- 50 Knoten ist das Systemlimit. So weit wie möglich von der weg Ressourcengrenze arbeiten.
- Es konzeptionell verstanden werden, wie ein Knoten bei DIVA.EXCHANGE hinzugefügt wird.

Studenten

- Tests durchführen
- Schriftliche Arbeit

7.1.4.21 KW 18 – 12. Besprechung

Alle

- Der erste Knoten ist wichtig für den Ping.
- Die 2/3 Mehrheit laut Spezifikationen benötigt im Netzwerk. Remove oder Add Peer brauchen 2/3 Mehrheit. Die Mehrheit ist nicht veränderbar.
- Crosschecking in YAC Whitepaper: supermajority > 2/3 -> 3f + 1
- Eventuell keine 16 registrierte Knoten. Genesis Block abändern oder per add peer arbeiten.

Studenten

- Tests durchführen
- Schriftliche Arbeit

7.1.4.22 KW 18 – 22. Besprechung

Alle

- Lib3 von Python ist einschränkend -> http connection pool.
- Unser Scope ist nicht Performance.

Studenten

- Tests durchführen
- Schriftliche Arbeit

7.1.4.23 KW 19 – 23. Besprechung

Alle

- Kein Meeting Donnertags.
- Umgehen des Add-Peer Problems dadurch, dass wir nur bis zur Grenze der Annahme eines Kommandos im Netzwerk gehen.

Studenten

- Tests durchführen
- Schriftliche Arbeit

7.1.4.24 KW 19 – 24. Besprechung

Kein Protokoll

7.1.4.25 KW 20 – 25. Besprechung

Alle

- Kein Meeting montags.
- Bereitstellen von möglichst grossem Teil der schriftlichen Arbeit zur Prüfung.

Studenten

- Schriftliche Arbeit

7.1.4.26 KW 20 – 26. Besprechung

Kein Protokoll

7.1.4.27 KW 21 – 27. Besprechung

Kein Protokoll

7.1.4.28 KW 21 – 28. Besprechung

Alle

- Korrektur von Stephan und dann neue Version an Carolyn und Konrad am Montagabend.
- Verteidigung am 2.7.2021 um 8:30 Uhr.

Studenten

- Letzter Schliff an der Arbeit mit den Korrekturen vornehmen.

7.1.4.29 KW 22 – 29. Besprechung

Kein Protokoll

7.1.4.30 KW 22 – 30. Besprechung

Kein Protokoll

7.2 Weiteres

In diesem letzten Abschnitt ist eine kurze Zusammenfassung des Codes für die Tests zu finden und die Resultate der Tests in Tabellenform. Weiter ist eine kurze Analyse inbegriffen, wie viel ein Angriff auf DIVA.EXCHANGE kosten könnte. Schlussendlich wird der Zweck und die Erkenntnisse aus den Helferskripten aufgezeigt. Die Helferskripte selbst sind aus Platzgründen nicht in dieser Arbeit angehängt. Mit einem Link bei jedem Abschnitt sind diese aber auf Codeberg einzusehen. Bei allen Skripten ist zudem in der ersten Zeile der Befehl vermerkt, mit dem man das Skript korrekt laufen lässt.

7.2.1 Tests Code

In diesem Abschnitt wird der Aufbau der Tests gezeigt und erläutert. Als erstes ist ein Klassendiagramm abgebildet, auf dem die Abhängigkeiten zu sehen sind. Danach wird der Inhalt des Setup-Moduls genauer gezeigt und abschliessend der Inhalt des Test-Moduls. Sämtlicher Code für die Tests ist mit Python implementiert und mit Python 3.8.5 getestet. Allfällige Docker- und System-Befehle sind mit "os.system()" implementiert und auf einer Ubuntu 20.04 VM getestet.

7.2.1.1 Klassendiagramm

In der folgenden Abbildung (33) ist das Setup-Modul und das Test-Modul abgebildet. Auf der folgenden Seite werden diese Module im Detail beschrieben.

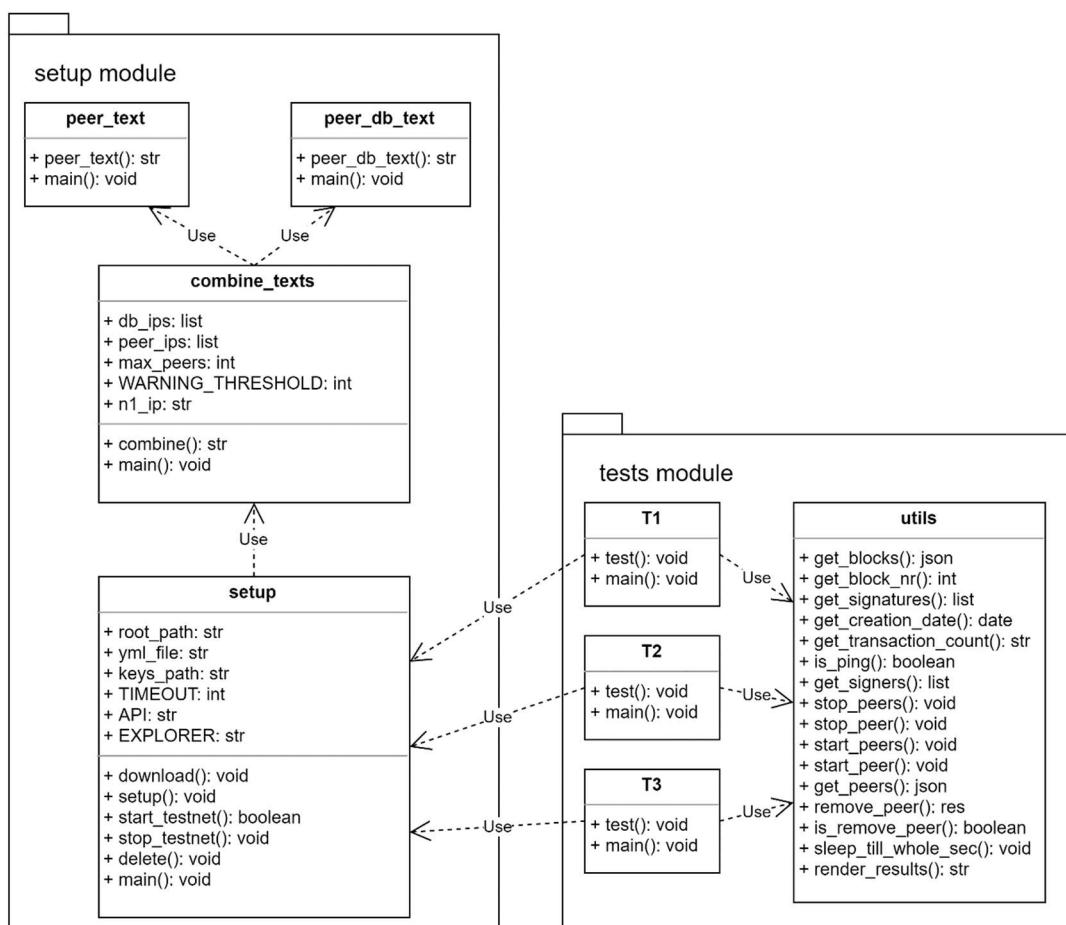


Abbildung 33: Klassendiagramm der Tests

7.2.1.2 Setup

Das Setup-Modul ist für die Konfiguration und das Aufstarten des Netzwerks verantwortlich. Dabei benutzt es das "combine_texts" Skript, um den neuen Inhalt der Konfigurationsdatei zu bestimmen. Dieses verwendet wiederum "peer_text" und "peer_db_text". Diese Skripte erstellen eine gültige Konfiguration für eine beliebige Anzahl von Knoten für das DIVA.EXCHANGE Testnetzwerk. Das "setup" Skript kann mit dieser Konfiguration dann selbst die Knoten erstellen und das Netzwerk starten.

Der Ablauf ist dabei wie folgt: Zuerst wird geprüft, ob die aktuellen Docker Container auf dem PC vorhanden sind. Sind diese nicht aktuell oder nicht vorhanden, werden diese von Docker Hub heruntergeladen. Darauf wird die Konfigurationsdatei für das neue Netzwerk erstellt. Mit dieser Konfiguration werden die Knoten erstellt, aber noch nicht gestartet. Zuerst muss der Genesis-Block angepasst werden, sodass alle neuen Knoten registriert sind, und alle öffentlichen Schlüssel der neuen Knoten müssen in alle Container kopiert werden. Ohne diese Schlüssel könnten sich die Knoten untereinander nicht verifizieren und es könnten keine Blöcke erstellt werden. Zusätzlich erhält jeder Knoten noch seinen privaten Schlüssel. Die Schlüssel sind einmal erstellt worden, für die beschriebenen Tests ist das ausreichend. Es existieren 100 Schlüsselpaare im Unterordner "/testing-diva/setup/keys/". Falls in Zukunft mehr gebraucht werden, oder für jeden Test neue Schlüssel benötigt werden, ist im selben Ordner noch das Skript vorhanden, dass diese 100 Schlüssel ursprünglich erstellt hatte.

Nachdem nun die Knoten angepasst sind, können diese gestartet werden. Dabei wird auf den Docker Prozess gewartet, bis alle Container laufen. Nach dem erfolgreichen Starten wird zusätzlich darauf gewartet, dass die Diva-API und der Blockchain Explorer online sind. Antworten diese auch, können die Tests beginnen.

7.2.1.3 Tests

Die Tests vom Test-Modul benutzen die Funktionalität vom "setup" Skript, um ein Netzwerk zu starten und die Tests durchzuführen. Die Funktionen aus dem "utils" Skript bieten den Tests Möglichkeiten mit dem Netzwerk zu interagieren oder Informationen aus der Blockchain zu extrahieren. Alle Tests haben eine "test" Methode, die für das Ausführen des Tests zuständig ist. Werden die Test Skripte selbst laufen gelassen (`if __name__ == "__main__"`), wird die Methode "test" aufgerufen, und per Kommandozeilenparameter kann zusätzlich angegeben werden, wie viele Knoten getestet werden sollen, oder ob der gesamte Test mit je 9, 15, 21, 27 und 33 Knoten gestartet werden soll. Werden keine Parameter mitgegeben, führen die Tests einen kompletten Durchgang mit 15 Knoten durch. Für allfällige gewünschte Anpassungen ist dieser Code der beste Einstiegspunkt, um die Funktionalität oder den Ablauf der Tests zu verändern.

Der Ablauf von allen drei Tests ist sehr ähnlich. Mit dem "setup" Skript wird ein Netzwerk erstellt und gestartet, und danach der Test durchgeführt. Am Ende des Tests werden die Resultate übersichtlich formatiert, in Dateien abgespeichert und auf dem Terminal ausgegeben. Wird während der Dauer des Tests ein Fehler geworfen, wird dieser aufgefangen, auf dem Terminal ausgegeben, die bestehenden Resultate gespeichert und das Netzwerk ordnungsgemäss heruntergefahren. In der neuesten Version der Tests sind keine Bugs mehr gefunden worden, somit ist diese Massnahme nur nützlich für zukünftige Entwickler dieser Tests.

7.2.2 Resultate – Tabellen

7.2.2.1 Tabellen für Test 1

Der Aufbau der abgebildeten Tabellen ist wie folgt: Der Titel der Tabelle bedeutet, wie viele Knoten, also Peers, im Netzwerk registriert sind. Die Spalte *ein* bedeutet, wie viele Knoten an einer Abstimmung teilnehmen. *Ping bei* bedeutet, nach wie vielen Sekunden der Ping in die Blockchain eingetragen worden ist, dies sollte normalerweise nach 60 Sekunden geschehen. Steht in dieser Spalte 900, ist ein Ping erwartet worden, aber es ist keiner angekommen. Steht in dieser Spalte 60 und in der Spalte *Sign.* ein "--" bedeutet das, dass kein Ping erwartet worden ist und auch keiner angekommen ist, aber sicherheitshalber trotzdem 60 Sekunde gewartet worden ist. Sonst stellt *Sign.* dar, wie viele Knoten den Ping validiert haben. Schlussendlich ist in der Spalte *Validatoren* ersichtlich, welche Knoten den Ping validiert haben. Ein "--" in der Spalte *Sign.* und *Validatoren* bedeutet, dass kein Ping angekommen ist.

9 Knoten				
ein	Ping bei	Sign.	Validatoren	
9	60	6	n1, n4, n3, n9, n2, n5	
8	54	6	n1, n8, n3, n2, n6, n4	
7	69	6	n1, n5, n6, n4, n7, n3	
6	99	5	n1, n5, n3, n6, n4	
5	107	5	n1, n5, n4, n3, n2	
4	60	--	--	
3	60	--	--	
2	60	--	--	
1	60	--	--	
0	60	--	--	

15 Knoten				
ein	Ping bei	Sign.	Validatoren	
15	65	8	n1, n7, n4, n11, n2, n3, n15, n6	
14	68	8	n1, n4, n3, n5, n2, n14, n13, n8	
13	66	9	n1, n10, n4, n6, n5, n9, n3, n8, n13	
12	105	8	n1, n4, n10, n5, n12, n2, n11, n8	
11	58	8	n1, n11, n9, n4, n7, n8, n10, n3	
10	76	8	n1, n4, n5, n7, n10, n3, n8, n6	
9	111	8	n1, n8, n4, n7, n3, n6, n5, n9	
8	256	8	n1, n2, n5, n4, n8, n7, n6, n3	
7	60	--	--	
6	60	--	--	
5	60	--	--	
4	60	--	--	
3	60	--	--	
2	60	--	--	
1	60	--	--	
0	60	--	--	

21 Knoten				
ein	Ping bei	Sign.	Validatoren	
21	60	11	n1, n2, n7, n18, n5, n21, n8, n16, n19, n13, n3	
20	57	12	n1, n12, n9, n18, n6, n13, n11, n5, n7, n19, n3, n10	
19	82	11	n5, n7, n2, n8, n14, n12, n11, n10, n18, n17, n15	
18	92	12	n1, n10, n5, n12, n18, n2, n14, n17, n15, n6, n9, n13	
17	90	12	n1, n12, n10, n6, n2, n14, n9, n13, n11, n5, n15, n17	
16	54	11	n1, n3, n16, n12, n11, n15, n9, n8, n6, n7, n2	
15	55	11	n1, n9, n5, n12, n7, n14, n13, n8, n11, n2, n3	
14	102	11	n1, n12, n7, n11, n8, n5, n9, n14, n6, n10, n13	
13	151	11	n1, n2, n12, n11, n9, n13, n8, n5, n3, n6, n10	
12	427	11	n1, n3, n5, n9, n12, n2, n11, n8, n4, n6, n10	
11	900	--	--	
10	60	--	--	
9	60	--	--	
8	60	--	--	
7	60	--	--	
6	60	--	--	
5	60	--	--	
4	60	--	--	
3	60	--	--	
2	60	--	--	
1	60	--	--	
0	60	--	--	

27 Knoten				
ein		Ping bei	Sign.	Validatoren
27		63	14	n1, n3, n5, n11, n18, n4, n16, n7, n12, n6, n8, n21, n27, n19
26		54	15	n1, n24, n18, n26, n16, n12, n7, n11, n5, n3, n23, n15, n8, n2, n9
25		224	14	n1, n18, n4, n11, n7, n3, n6, n23, n5, n8, n20, n21, n12, n22
24		101	15	n1, n11, n3, n4, n22, n23, n19, n8, n18, n7, n17, n5, n10, n14, n20
23		111	14	n1, n3, n11, n22, n23, n18, n19, n9, n20, n2, n6, n21, n7, n12
22		55	14	n1, n8, n22, n11, n3, n9, n16, n18, n19, n20, n12, n6, n10, n14
21		103	14	n1, n18, n3, n14, n5, n19, n10, n8, n2, n6, n9, n21, n15, n13
20		103	14	n1, n15, n5, n8, n18, n3, n11, n9, n2, n19, n6, n12, n14, n20
19		60	14	n1, n10, n11, n8, n18, n5, n3, n16, n6, n9, n14, n19, n15, n17
18		57	14	n1, n3, n11, n18, n16, n5, n15, n6, n2, n10, n9, n8, n14, n17
17		116	14	n1, n4, n8, n7, n6, n3, n13, n2, n11, n16, n5, n12, n9, n10
16		900	--	--
15		900	--	--
14		900	--	--
13		60	--	--
12		60	--	--
11		60	--	--
10		60	--	--
9		60	--	--
8		60	--	--
7		60	--	--
6		60	--	--
5		60	--	--
4		60	--	--
3		60	--	--
2		60	--	--
1		60	--	--
0		60	--	--

33 Knoten				
ein	Ping bei	Sign.	Validatoren	
33	80	17	n19, n22, n2, n27, n20, n17, n16, n28, n5, n18, n33, n8, n15, n21, n26, n4, n11	
32	58	18	n1, n20, n30, n22, n27, n19, n2, n16, n17, n26, n25, n12, n21, n4, n5, n28, n32, n15	
31	55	17	n22, n16, n20, n2, n12, n26, n21, n5, n4, n24, n29, n28, n31, n15, n6, n18, n10	
30	60	17	n1, n23, n21, n2, n14, n19, n27, n20, n8, n18, n10, n22, n28, n16, n17, n3, n30	
29	58	18	n1, n2, n19, n12, n26, n21, n14, n18, n5, n16, n28, n25, n10, n6, n24, n7, n17, n3	
28	62	17	n1, n21, n6, n26, n12, n5, n24, n18, n9, n2, n14, n3, n19, n13, n4, n23, n15	
27	82	17	n1, n22, n16, n27, n17, n21, n3, n9, n7, n11, n18, n8, n10, n4, n26, n12, n23	
26	98	18	n1, n15, n22, n17, n19, n2, n26, n16, n5, n6, n25, n14, n8, n4, n24, n18, n10, n20	
25	59	18	n1, n22, n19, n17, n16, n23, n25, n24, n12, n15, n18, n11, n4, n5, n6, n21, n2, n3	
24	106	18	n1, n12, n21, n24, n9, n15, n14, n16, n2, n4, n13, n7, n17, n11, n19, n18, n5, n10	
23	85	18	n1, n22, n16, n4, n17, n19, n5, n2, n7, n13, n20, n18, n10, n6, n12, n8, n14, n3	
22	56	18	n1, n19, n17, n22, n16, n18, n11, n14, n7, n10, n21, n2, n8, n4, n12, n13, n9, n3	
21	54	17	n1, n19, n16, n2, n17, n18, n7, n5, n10, n4, n13, n8, n11, n12, n6, n9, n14	
20	97	18	n1, n12, n6, n14, n15, n9, n3, n20, n19, n2, n4, n7, n17, n5, n16, n11, n13, n18	
19	900	--	--	
18	900	--	--	
17	900	--	--	
16	60	--	--	
15	60	--	--	
14	60	--	--	
13	60	--	--	
12	60	--	--	
11	60	--	--	
10	60	--	--	
9	60	--	--	
8	60	--	--	
7	60	--	--	
6	60	--	--	
5	60	--	--	
4	60	--	--	
3	60	--	--	
2	60	--	--	
1	60	--	--	
0	60	--	--	

7.2.2.2 Tabellen für Test 2

Der Aufbau der abgebildeten Tabellen ist wie folgt: Der Titel der Tabelle bedeutet, wie viele Knoten, also Peers, im Netzwerk registriert sind. Die Spalte *ein* bedeutet, wie viele Knoten an einer Abstimmung teilnehmen. *entfernt* zeigt an, ob und nach wie vielen Sekunden ein Knoten aus dem Netzwerk entfernt worden ist. Steht in dieser Spalte 900, ist erwartet worden, dass der Knoten entfernt werden kann, allerdings ist der Befehl in dieser Zeit nicht durchgekommen. Steht in dieser Spalte 60 und in der Spalte *Sign.* ein "--" bedeutet das, dass ein erfolgreiches Entfernen nicht erwartet worden ist und auch nichts passiert ist, aber sicherheitshalber trotzdem 60 Sekunde gewartet worden ist. Sonst stellt *Sign.* dar, wie viele Knoten das Entfernen validiert haben. Schlussendlich ist in der Spalte *Validatoren* ersichtlich, welche Knoten das Entfernen validiert haben. Ein "--" in der Spalte *Sign.* und *Validatoren* bedeutet, dass das Entfernen erfolglos gewesen ist.

Eine grüne Zeile bedeutet, dass das Verhältnis vorteilhafter wäre für die Angreifer, und somit ein Test unnötig ist. Eine rote Zeile bedeutet, dass auch mit $2f+1$ ein Knoten nicht entfernt werden kann, und zudem die Performance so schlecht ist, dass das nicht getestet wurde.

9 Knoten					
ein		entfernt	nach	Sign.	Validatoren
1		--	60	--	--
2		--	60	--	--
3		--	60	--	--
4		--	60	--	--
5		n9	131	5	n1, n4, n5, n2, n3
6		--	--	--	--
7		--	--	--	--
8		--	--	--	--
9		--	--	--	--

15 Knoten					
ein		entfernt	nach	Sign.	Validatoren
1		--	60	--	--
...	
6		--	60	--	--
7		--	60	--	--
8		n15	24	8	n1, n8, n7, n5, n2, n4, n6, n3
9		--	--	--	--
...	
15		--	--	--	--

21 Knoten					
ein		entfernt	nach	Sign.	Validatoren
1		--	60	--	--
...	
9		--	60	--	--
10		--	60	--	--
11		--	900	--	--
12		--	900	--	--
13		n21	82	11	n4, n11, n10, n13, n12, n9, n8, n2, n5, n7, n6
14		--	--	--	--
...	
21		--	--	--	--

27 Knoten					
ein		entfernt	nach	Sign.	Validatoren
1		--	60	--	--
...	
12		--	60	--	--
13		--	60	--	--
14		--	900	--	--
15		n27	47	15	n1, n14, n13, n15, n11, n4, n2, n8, n3, n6, n12, n9, n10, n5, n7
16		--	--	--	--
...	
27		--	--	--	--

33 Knoten					
ein		entfernt	nach	Sign.	Validatoren
1		--	60	--	--
...	
15		--	60	--	--
16		--	60	--	--
17		--	900	--	--
18		--	900	--	--
19		n33	161	17	n1, n6, n2, n8, n17, n18, n7, n12, n11, n16, n3, n19, n9, n13, n10, n5, n4
20		--	--	--	--
...	
33		--	--	--	--

7.2.2.3 Tabellen für Test 3

Der Aufbau der abgebildeten Tabellen ist wie folgt: Der Titel der Tabelle bedeutet, wie viele Knoten, also Peers, im Netzwerk registriert sind. Die Spalte *ein* bedeutet, wie viele Knoten an einer Abstimmung teilnehmen. *Ping bei* bedeutet, nach wie vielen Sekunden der Ping in die Blockchain eingetragen worden ist, dies sollte normalerweise nach 60 Sekunden geschehen. Steht in dieser Spalte 900, ist ein Ping erwartet worden, aber es ist keiner angekommen. Steht in dieser Spalte 60 und in der Spalte *Sign.* ein "--" bedeutet das, dass kein Ping erwartet worden ist und auch keiner angekommen ist, aber sicherheitshalber trotzdem 60 Sekunde gewartet worden ist. Sonst stellt *Sign.* dar, wie viele Knoten den Ping validiert haben. Schlussendlich ist in der Spalte *Validatoren* ersichtlich, welche Knoten die Ping validiert haben. Ein "--" in der Spalte *Sign.* und *Validatoren* bedeutet, dass kein Ping angekommen ist.

Eine rote Zeile bedeutet, dass auch mit $2f+1$ ein Knoten nicht entfernt werden kann, und zudem die Performance so schlecht ist, dass das nicht getestet worden ist.

9 Knoten				
ein		Ping bei	Sign.	Validatoren
1		60	--	--
2		60	--	--
3		60	--	--
4		60	--	--
5		140	5	n1, n5, n4, n3, n2
6		99	5	n1, n3, n4, n5, n2
7		155	5	n7, n2, n4, n5, n6
8		113	6	n1, n7, n3, n4, n5, n2
9		70	6	n1, n8, n9, n5, n3, n7

15 Knoten				
ein		Ping bei	Sign.	Validatoren
1		60	--	--
...	
6		60	--	--
7		60	--	--
8		43	8	n1, n8, n7, n5, n4, n2, n3, n6
9		156	8	n1, n3, n4, n9, n5, n7, n2, n8
10		109	9	n1, n2, n8, n10, n7, n6, n9, n5, n3
11		109	8	n1, n11, n3, n8, n9, n2, n4, n10
12		152	8	n1, n12, n4, n7, n2, n9, n8, n11
13		64	9	n1, n11, n5, n7, n13, n6, n4, n8, n2
14		68	9	n1, n2, n3, n13, n9, n6, n14, n11, n12
15		70	8	n14, n6, n8, n13, n3, n12, n4, n2

21 Knoten				
ein		Ping bei	Sign.	Validatoren
1		60	--	--
...	
9		60	--	--
10		60	--	--
11		900	--	--
12		900	--	--
13		187	12	n1, n8, n7, n4, n13, n9, n3, n5, n12, n11, n2, n10
14		102	11	n1, n12, n3, n5, n14, n4, n11, n9, n2, n10, n6
15		104	11	n1, n15, n13, n8, n7, n14, n3, n9, n5, n6, n11
16		163	11	n9, n11, n13, n2, n3, n10, n16, n8, n7, n4, n15
17		113	11	n1, n13, n11, n7, n5, n8, n15, n4, n14, n10, n9
18		100	12	n1, n17, n18, n5, n8, n16, n4, n10, n7, n3, n11, n15
19		118	11	n19, n13, n3, n8, n7, n18, n10, n17, n5, n12, n2
20		112	12	n1, n20, n18, n12, n9, n15, n3, n16, n14, n17, n13, n5
21		90	11	n1, n18, n12, n20, n2, n7, n15, n17, n3, n19, n10

27 Knoten				
ein		Ping bei	Sign.	Validatoren
1		60	--	--
...	
12		60	--	--
13		60	--	--
14		900	--	--
15		108	14	n1, n14, n13, n2, n6, n9, n10, n11, n12, n7, n8, n3, n5, n4 n1, n16, n14, n13, n5, n8, n10, n3, n11, n6, n12, n15, n4,
16		116	14	n7
17		103	14	n1, n4, n5, n7, n15, n12, n11, n6, n10, n2, n8, n3, n14, n13
18		114	15	n1, n3, n18, n16, n13, n11, n7, n12, n17, n14, n6, n9, n2, n10, n4
19		163	14	n1, n19, n3, n11, n8, n18, n5, n16, n10, n6, n17, n4, n13, n15
20		63	15	n1, n20, n7, n16, n4, n5, n8, n15, n19, n18, n17, n2, n14, n10, n13
21		91	15	n1, n5, n16, n7, n4, n8, n19, n3, n9, n15, n13, n12, n20, n10, n18
22		110	15	n1, n22, n3, n18, n5, n16, n12, n9, n11, n21, n8, n7, n19, n13, n10
23		80	15	n1, n15, n2, n23, n14, n4, n20, n12, n19, n18, n21, n13, n9, n7, n17
24		123	14	n1, n24, n11, n18, n4, n21, n23, n20, n13, n10, n2, n22, n15, n9
25		90	15	n1, n14, n25, n8, n7, n17, n6, n13, n24, n12, n5, n21, n23, n3, n16
26		123	14	n1, n22, n8, n4, n7, n25, n21, n18, n24, n17, n15, n5, n16, n14
27		62	15	n1, n3, n26, n11, n12, n24, n19, n25, n18, n5, n21, n9, n15, n16, n8

33 Knoten				
ein		Ping bei	Sign.	Validatoren
1		60	--	--
...	
15		60	--	--
16		60	--	--
17		900	--	--
18		900	--	--
19		900	--	--
20		55	18	n1, n19, n17, n18, n16, n9, n3, n10, n6, n4, n5, n7, n12, n13, n8, n2, n11, n15
21		148	17	n21, n19, n6, n12, n5, n14, n16, n13, n10, n11, n17, n7, n4, n15, n9, n2, n3
22		71	17	n1, n7, n6, n16, n9, n13, n18, n20, n12, n10, n15, n2, n14, n11, n21, n3, n8
23		61	18	n1, n22, n21, n19, n17, n16, n12, n2, n6, n14, n9, n5, n15, n18, n8, n10, n11, n4
24		115	18	n1, n24, n22, n2, n5, n19, n16, n6, n4, n12, n18, n14, n11, n17, n21, n9, n13, n15
25		165	18	n1, n25, n19, n22, n12, n16, n14, n3, n21, n24, n20, n18, n5, n7, n8, n10, n17, n2
26		164	19	n1, n26, n19, n17, n5, n12, n6, n25, n23, n11, n21, n18, n2, n13, n10, n3, n7, n4, n24
27		63	18	n1, n27, n22, n19, n17, n2, n18, n23, n7, n4, n6, n9, n13, n21, n5, n10, n20
28		104	18	n1, n17, n25, n21, n2, n26, n24, n5, n6, n7, n4, n19, n16, n22, n14, n23, n28, n13
29		113	17	n22, n5, n27, n6, n2, n7, n19, n24, n4, n18, n15, n20, n3, n8, n9, n10, n13
30		81	17	n1, n30, n22, n19, n27, n25, n16, n12, n17, n20, n5, n3, n26, n24, n8, n29, n23
31		65	18	n1, n22, n19, n27, n26, n25, n18, n10, n9, n20, n8, n3, n6, n24, n16, n7, n21, n4
32		63	17	n1, n26, n5, n2, n21, n12, n22, n30, n24, n20, n29, n16, n25, n4, n7, n6, n27
33		69	17	n1, n30, n22, n29, n20, n24, n10, n8, n31, n17, n11, n9, n23, n4, n13, n5, n32

7.2.3 Kosten für 51 % Angriff auf Iroha Blockchain

Im folgenden Abschnitt wird ermittelt, wie viel ein Angriff auf ein DIVA.EXCHANGE Netzwerk mit 1000 Knoten kosten könnte. Damit der Angreifer erfolgreich beliebig neue Blöcke hinzufügen kann oder das Hinzufügen von neuen Blöcken in die Blockchain verhindert, muss er mit der aktuellen Implementation von DIVA.EXCHANGE und Iroha mehr als die Hälfte aller Knoten im Netzwerk kontrollieren, siehe Kapitel Resultate. Allerdings ist in den Resultaten auch ersichtlich, dass wenn der Angreifer weniger als 60 % der Knoten kontrolliert, das Netzwerk sehr langsam und unverlässlich sein kann. Folglich sollte ein Angreifer bei einem Netzwerk mit 1000 bestehenden Knoten 1500 neue Knoten erstellen, damit er das Verhältnis von 60 % erreicht. Zusätzlich wird angenommen, dass ein 2.5 mal grösseres Netzwerk, also eines mit 2500 Knoten, überhaupt noch funktioniert.

Zwei Möglichkeiten für einen Angriff werden folgend analysiert. Die erste ist mithilfe dem Cloud Computing Anbieter Linode. Bei Linode wird IaaS angeboten, also "Infrastructure as a Service". Kurz gesagt lässt sich damit ein "PC" mieten, mit dem man über ein Terminal kommunizieren kann. Eine graphische Oberfläche existiert nicht, trotzdem können Skripte und Programme normal ausgeführt werden, wie zum Beispiel die gezeigten Test Skripte.

Als zweite Möglichkeit könnte ein Angreifer ein C&C Botnet mieten oder kaufen. Ein Botnet besteht meist aus IoT Geräten, also "Internet of Things" Geräte, wie zum Beispiel ein Kühlschrank oder Toaster mit Internetanschluss. Diese Geräte besitzen meist sehr niedrige Sicherheitsvorkehrungen gegen externe Angriffe und sind somit leicht zu kontrollieren von unautorisierten Benutzern, also Angreifer [46]. C&C bedeutet "Command and Control", also dass der "Mieter" / "Besitzer" des Botnets mehr oder weniger selbst bestimmen kann, was die Bots machen. Im Vergleich dazu gibt es DDoS Botnetze, bei denen der "Mieter" / "Besitzer" nur die IP oder Domain angeben kann, aber nicht das genaue Verhalten der Bots, wie zum Beispiel beim Mirai Botnet [47].

Der Vorteil von Linode in diesem Szenario wäre die Verlässlichkeit und der Preis. Der Vorteil bei einem Botnet wäre die besser geschützte Anonymität. Obwohl bei Linode das Bezahlen mit Prepaid-Kreditkarten möglich ist, bieten diese nur einen kleinen Schutz der Anonymität im Vergleich zu einer Zahlung an einen "Botnet-Vermieter" mit einer Kryptowährung wie zum Beispiel Monero.

Zu den benötigten Ressourcen bei Linode; Für 50 Knoten werden auf einer Ubuntu VM ungefähr 13 GB RAM benötigt. Um 1500 Knoten zu starten sind somit etwa 390 GB RAM benötigt. Mit etwas Reserve sind das 400 GB RAM bei einem anzugreifenden Netzwerk von 1000 Knoten. 300GB sind bei Linode für 960\$ pro Monat oder 1.44\$ pro Stunde erhältlich [48]. Pro 1000 Knoten im Netzwerk würde somit ein Angriff über Linode 1280\$ pro Monat oder **1.92\$ pro Stunde** kosten.

Bei einem Botnet ist es einfacher umzusetzen, dass jeder Bot nur einen Knoten im Netzwerk ist. Somit sind auch ein eventuelles Entdecken und Verhindern des Angriffs erheblich schwieriger, weil die Bots von vielen unterschiedlichen Orten zugreifen und schwerer oder gar nicht von legitimen Knoten zu unterscheiden sind. Im Moment sind sowieso noch keine Verteidigungen bei DIVA.EXCHANGE oder Iroha dazu eingebaut. Falls diese aber kommen, sollte ein Botnet potenziell immer noch erfolgreich sein, im Vergleich zu einer grossen VM die bis zu 1150 Knoten beinhalten kann (300 GB RAM / 13 GB RAM pro 50 Knoten), weil diese leichter als eine Einheit zu erkennen ist.

Die Datenlage, wie viel ein Botnet kostet, ist sehr schmal. Ein Beitrag zeigt auf, dass für ein Botnet mit 1000 Bots 25\$ gefordert wird [49]. Es ist nicht ersichtlich, ob der Betrag einmalig oder pro Zeiteinheit ist. Es wird angenommen, dass dieser einmalig ist. Das Botnet bleibt dann auf ungewisse Dauer bestehen. Somit kostet ein Angriff bei einem Netzwerk mit 1000 Knoten durchschnittlich **37.5\$**. Voraussichtlich kann aber keine Garantie gegeben werden, wie lange die 1000 Knoten effektiv verfügbar sind.

7.2.4 Helferskripte

Folgend werden alle Skripte aufgelistet, die in der Arbeit verwiesen sind. Die Funktionsweise wird kurz erläutert, das Resultat erklärt und ein Link zum originalen Skript ist vermerkt.

7.2.4.1 Differenz 2f+1 zu 3f+1

Skript: https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/2f_3f/2f_3f.py

Mithilfe von diesem Skript kann gezeigt werden, wie viele Knoten für 2f+1 und 3f+1 bei einer gegebenen Netzwerkgrösse eingeschalten sein müssen. Beim Starten des Skripts kann angegeben werden, bis zu welcher Netzwerkgrösse gerechnet werden soll. Wird z.B. 50 angegeben, werden für alle Netzwerkgrößen von 1, 2, ... bis 50 die benötigten Knoten für 2f+1 und 3f+1 ausgegeben. Zusätzlich kann beim Starten angegeben werden, ob nur die Resultate bei optimalen Netzwerkgrößen oder alle Resultate ausgegeben werden sollen. Als Beispiel, wird bis zu 50 Knoten gerechnet und nur optimale Größen, ist das Resultat:

Knoten	2f + 1	3f + 1	diff
1	1	1	0
3	2	3	1
9	5	7	2
15	8	11	3
21	11	15	4
27	14	19	5
33	17	23	6
39	20	27	7
45	23	31	8

Das heisst bei einer Netzwerkgrösse von drei Knoten müssen für 2f+1 zwei Knoten eingeschalten sein und für 3f+1 müssen 3 eingeschalten sein. Die Differenz ist somit ein Knoten. Bei einer Netzwerkgrösse von 45 Knoten müssen für 2f+1 23 Knoten eingeschalten sein, und für 3f+1 31. Das entspricht einer Differenz von 8.

Wenn die Ausgabe von allen Resultaten aktiviert ist, liefert das Skript die folgende Tabelle:

Knoten	2f + 1	3f + 1	diff
1	1	1	0
2	2	2	0
3	2	3	1
4	3	3	0
5	3	4	1
6	4	5	1
7	4	5	1
8	5	6	1
9	5	7	2
10	6	7	1
...			
49	25	33	8
50	26	34	8

7.2.4.2 Docker Ressourcenverbrauch

Skript: https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/benchmark/system_resources.py

Dieses Skript startet wie der normale Teststand ein Netzwerk mit einer vorgegebenen Anzahl an Knoten. Bei 50 Knoten kann das Netzwerk gerade noch starten, bei 60 und 70 hängt der PC. Es ist ersichtlich, dass bei 50 Knoten beim Starten der Docker-Container ein Grossteil der RAMs verbraucht ist. Bei 60 und 70 Knoten geht der Verbrauch nur bis etwa in die Hälfte des verfügbaren RAMs, aber Docker startet auch nur etwa die Hälfte der Knoten. Die Schlussfolgerung davon ist, dass Docker keine neuen Container startet, wenn es zu wenig Speicher haben könnte. In diesem Moment wäre noch Speicher im RAM vorhanden, um einige Container zu starten, aber voraussichtlich nicht genug für alle Container.

Während diesem Prozess, also während Docker alle Container am Aufstarten ist, wird in einem separaten Thread der folgende Befehl ausgeführt:

```
top -b -n 100 -d 1 | grep \"top - \" -A5 > benchmark/system_resources_results.txt
```

"top" liest 100 Sekunden jede Sekunde den aktuellen Ressourcenverbrauch des PCs. Mit "grep" werden nur die relevantesten Attribute behalten und diese in das File "system_resources_results.txt" gespeichert. Die Resultate sind unter folgendem Link einzusehen:

https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/benchmark/system_resources_results.txt

Aus den Resultaten ist ersichtlich, dass beim getesteten PC die Dauer des Aufstartens nur von der CPU abhängt. Ob die Container vollständig starten, und somit bereit für einen Angriff wären, hängt von dem verfügbaren RAM ab. Wie in der Diskussion erwähnt wird bei der aktuellen Version der Iroha Container damit gerechnet, dass für 50 Container 13 GB RAM benötigt werden.¹ Die CPU und der Festplattenspeicher ist nur minim relevant, und diese sind bei Online-Diensten wie Linode im Vergleich zum RAM immer in Überschuss vorhanden [48].

¹ Es ist mit dem Betriebssystem Ubuntu 20.04 in einer VM von VMware Player 15 getestet worden. Auf dem Host läuft Windows 10.0.19042. Der Host hat insgesamt 16 GB RAM eingebaut, die VM ist in einer SSD mit 500GB Speicher platziert und die CPU des Hosts hat 4 Kerne und 4 logische Prozessoren mit je 3.5 GHz.

7.2.4.3 Docker Zugriffsrechte

Skript: https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/docker_access/inspect.sh

Das obenstehende Skript startet einen einzelnen Iroha Container und listet die Rechte der einzelnen Dateien im Container auf. Wie im Kapitel 4.1 Statische Analyse aufgezeigt wird, haben alle Dateien im Container die folgenden Rechte:

```
-rw-rw-r-- 1 root root
```

Theoretisch kann also nur der root User die Dateien anpassen. Jeder Benutzer sollte aber die Dateien lesen können. Verhindert wird das Lesen, weil nur der root User Leserechte auf den Docker Ordner hat, welcher alle Dateien beinhaltet, bei Ubuntu 20.04 unter "/var/lib/docker".

Ab der Zeile 40 im Skript ist zusätzlich ersichtlich, was geschieht, wenn ein normaler User die Dateien im Container versucht zu lesen. Wie erwartet, kann ein normaler User die Dateien nicht über das normale Filesystem des Betriebssystems lesen. Ebenfalls wird dem normalen User den Zugriff auf den Docker-Prozess verweigert, mit dem man ebenfalls die Dateien auslesen könnte.

Dies alles kann umgangen werden, wenn ein normaler Benutzer unautorisiert root Rechte erhält, mithilfe sogenannter privilege escalation in anderer Software. Allerdings kann Iroha und DIVA.EXCHANGE dagegen nichts unternehmen, und es liegt in der Verantwortung des Benutzers, welche Software sonst installiert ist auf dem PC.

7.2.4.4 Beispiele für Rechte an Dateien

Skript: https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/docker_access/file_rights_example.sh

Mit dem verlinkten Skript wird eine Datei mit folgenden Rechten untersucht:

```
-rw-rw-r-- 1 root root
```

Die Rechte sind somit genau gleich wie beim vorherigen Abschnitt, "Docker Zugriffsrechte". In der Datei "file_rights_example_output.txt" sind die Ergebnisse der Untersuchung zu sehen. Will man die Untersuchung selbst durchführen, muss nur die Datei "file_rights_example.sh" laufen gelassen werden.

Ersichtlich wird, dass ein normaler Benutzer, der nicht root ist, und nicht in der root Gruppe ist, die Datei nur lesen kann. Im Vergleich zum vorherigen Abschnitt haben alle Benutzer Rechte, um die Inhalte des enthaltenden Ordners zu lesen. Somit kann im vorherigen Abschnitt gezeigt werden, dass doch nur der root User Zugriff auf die Schlüssel hat.

7.2.4.5 Wahrscheinlichkeit für korrekte Auswahl von Abstimmungsknoten

Skript: https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/consensus_shuffle/correct_shuffle_chance.py

Mit dem obenstehenden Skript kann ausgerechnet werden, wie gross die Chance ist, eine korrekt Auswahl zu generieren. Für eine gültige Auswahl müssen die ausgewählten Knoten auf die Abstimmung antworten. Sind zum Beispiel Knoten 1 und 3 ausgeschaltet und Knoten 2, 4 und 5 eingeschaltet, wären nur (2, 4, 5), (2, 5, 4), (4, 2, 5), (4, 5, 2), (5, 2, 4) und (5, 4, 2) gültige Auswahlen. Auswahlen, die den Knoten 1 oder 3 enthalten, wären in dieser Situation nicht gültig, weil nicht genügend Knoten auf die Abstimmung antworten. In dieser Situation sind 120 unterschiedliche Auswahlen möglich ($5! = 120$). Nur sechs davon sind gültig, somit ist durchschnittlich nur jede 20ste Auswahl gültig. Mit einer grösseren Anzahl an Knoten wächst diese Zahl exponentiell.

Wie schon in den Resultaten erwähnt, ist nicht sicher, ob Iroha diesen simplen Algorithmus zur Auswahl von abstimgenden Knoten exakt so implementiert. Festzustellen ist aber, dass die Performance von Iroha mit steigender Netzwerkgrösse und sinkender Anzahl an eingestellten Knoten stark abnimmt. Daraus lässt sich schliessen, dass Iroha einen unpassenden Algorithmus implementiert, der ähnlich funktioniert wie der im vorherigen Abschnitt beschriebene.

7.2.4.6 Dauer für eine Entfernung bei einem vollständig aktiven Netzwerk

Skript: <https://codeberg.org/DIVA.EXCHANGE/BA2021-Caillev-Kybursas/src/branch/master/testing-diva/benchmark/remove.py>

Das obenstehende Skript erstellt ein Netzwerk mit einer gewissen Anzahl an Knoten. Nach dem erfolgreichen Starten der Knoten wird eine Anfrage zur Entfernung eines Knoten erstellt. Im selben Ordner wie das Skript sind auch die Resultate von einer Entfernung mit 9 Knoten und 33 Knoten insgesamt enthalten.

Das Skript bestätigt, dass eine normale Entfernung eines Knoten zwischen 5 und 10 Sekunden dauern kann. Allerdings ist auch bei einer Entfernung eine Abstimmung nötig. Deshalb gilt auch beim Entfernen, dass die Performance von Iroha eine Rolle spielt. Und erneut ist die Performance von Iroha relativ schlecht, wenn nur etwas mehr als die Hälfte der Knoten eingeschaltet ist. Wie in den Resultaten ersichtlich, muss das Verhältnis von eingeschalteten Knoten zu Knoten insgesamt etwa 0.6 betragen, damit ein Knoten mehr oder weniger verlässlich entfernt werden kann.