Name: Tobias Binnewies

Hochschule Weserbergland
Studiengang: Wirtschaftsinformatik
Studiengruppe: WI67/21
Dozent: Ralf Hesse

Lösungsorientierte Transferarbeit 2 für Semester 5
(Zeitraum vom 04.12.2023 bis 31.01.2024)

Thema:
Eignungsanalyse von Distributed Ledger Systemen in der Finanz Informatik GmbH & Co. KG

Praxispartner
Finanz Informatik GmbH & Co. KG
Laatzener Straße 5, 30539 Hannover

I Inhaltsverzeichnis

II Abkürzungsverzeichnis

DL	Distributed Ledger
DLS	Distributed Ledger System
DSGVO	Datenschutz-Grundverordnung
ERC	Ethereum Request for Comments
EIP	Ethereum Improvement Proposal
ETH	Ether (Währung)
EVM	Ethereum Virtual Machine
FI	Finanz Informatik GmbH & Co. KG
HD	Hierarchical Deterministic
OSPE	One System Plus Enterprise
OSPlus / OSP	One System Plus
PoW	Proof of Work
PoS	Proof of Stake
PoA	Proof of Authority
TPS	Transactions per Second
UTXO	Unspent Transaction Output
ZK	Zero Knowledge

III Abbildungsverzeichnis

1 Einleitung

In dieser Arbeit wird auf die Eignung bzw. auf Use Cases von Distributed Ledger Systemen in der Finanz Informatik eingegangen. Speziell wird dabei der Zahlungsverkehr betrachtet. Dieser ist das Kerngeschäft der Sparkassen und so ein wichtiger Teil der FI. Aktuell wird für diesen ein Großrechner verwendet. Allerdings gibt es schon seit vielen Jahren das Ziel, diesen abzuschaffen und durch eine andere Technologie zu ersetzten. Daher könnte ein DLS eine Alternative darstellen.
Die Arbeit befindet sich im Scope der FI. Es wird also nicht der gesamte Zahlungsverkehr betrachtet, sondern nur die konkrete Implementierung des Zahlungsverkehrs in der FI.
Zunächst werden DLS im allgemeinen betrachtet und erklärt, um ein Grundverständnis für diese Technologie zu schaffen. Dann folgt eine Betrachtung des Ist-Zusands des Zahlungsverkehrs, um die Anforderungen an diesen zu ermitteln. Nachfolgend werden Möglichkeiten der Verwendung von DLS im Zahlungsverkehr betrachtet. Abschließend folgt eine Schlussfolgerung und ein Ausblick auf weitere mögliche Verwendungsmöglichkeiten von DLS.

2 Distributed Ledger System

2.1 Distributed Ledger

Ein Distributed Ledger ist eine Datenbank, die im Konsens geteilt und über ein Netzwerk synchronisiert wird, das sich über mehrere Standorte, Institutionen oder Länder erstreckt.[1] Es ermöglicht, dass Transaktionen und Aufzeichnungen öffentlich und überprüfbar sind, und da es dezentralisiert ist, gibt es keinen einzelnen Ausfallpunkt. Jeder Teilnehmer im Netzwerk hat Zugang zu den Aufzeichnungen, die über dieses Netzwerk geteilt werden und kann eine identische Kopie der Daten haben. Änderungen oder Ergänzungen am DL werden nahezu in Echtzeit in allen Kopien widergespiegelt, was die Transparenz und Sicherheit erhöht.

Ein Distributed Ledger System ist ein System, das einen Distributed Ledger verwendet, um Transaktionen zwischen Teilnehmern zu verwalten. Am häufigsten wird die Blockchain-Technologie als DL verwendet.[2]

2.2 Blockchain

Eine Blockchain ist eine spezielle Form eines Distributed Ledgers, die aus einer Kette von Blöcken besteht, die jeweils die zu speichernden Daten enthalten.[3] Ein Block

[1] Vgl. hierzu und zum Folgenden LedgerAcademy (2023); Majaski (2023).
[2] Vgl. Tamalio (2023).
[3] Vgl. Greeh, Camilleri (2017), S. 16.

besteht aus einem Header und einem Body.[4]

Der Header enthält u. a. den Hash des vorherigen Blocks, einen Zeitstempel und die Nummer des Blocks. Außerdem enthält der Header einen Hash des Bodys. So wird gewährleistet, dass die Blöcke - und so auch die darin beinhalteten Daten - nach ihrem Eintrag nicht verändert werden können, ohne alle nachfolgenden Blöcke abzuändern. (So wird von „Block-Confirmation" gesprochen, wenn eine bestimmte Anzahl von Blöcken nach diesem Block hinzugefügt wurden - je mehr desto sicherer -, da er erst dann als „unveränderlich" gilt.[5])

Der Body enthält die eigentlichen Daten, die gespeichert werden sollen. Im Falle einer Kryptowährung sind dies bspw. die Transaktionen, die in diesem Block gespeichert werden.

2.3 Kryptowährung

Ein DLS kann viele Anwendungsmöglichkeiten haben. Am wohl bekanntesten ist die Verwendung als Kryptowährung, wie bspw. Bitcoin.[6] Hierbei werden die Transaktionen zwischen den Teilnehmern des Netzwerks durchgeführt, die in einer Blockchain festgehalten werden. Da die Korrektheit der Transaktionen von allen Teilnehmern geprüft werden, können Transaktionen Peer-to-Peer durchgeführt werden, also ohne dass eine zentrale Instanz diese überprüfen muss. Es muss also kein Vertrauen in eine zentrale Instanz gesetzt werden.

2.4 Ethereum Virtual Maschine Chain

EVM-Chains heben das Konzept DLS auf eine neue Ebene, indem sie es ermöglichen, nicht nur Transaktionen abzuwickeln, sondern dazu noch vorprogrammierten Code (Smart Contracts) auszuführen und dadurch viele neue Anwendungsmöglichkeiten erschaffen.[7] Eine EVM-Chain ist also ein Blockchain-Netzwerk, das die Ethereum Virtual Machine (EVM) verwendet, um Smart Contracts auszuführen. Ethereum selbst ist eine EVM-Chain, die die Kryptowährung Ether (ETH) verwendet. Allerdings gibt es noch weitere Chains, die ebenfalls kompatibel mit der EVM sind, wie bspw. Binance Smart Chain (BSC) oder Polygon (MATIC)[8]. So kann für jede dieser Chains der gleiche Code - sowie Tools für dessen Entwicklung - verwendet werden, um Smart Contracts zu erstellen, die dann auf der jeweiligen Chain ausgeführt werden können.

[4] Vgl. hierzu und zum Folgenden Singhal, Dhameja, Panda (2018), S. 161 ff.; Ethereum.org (2023a).
[5] Vgl. Singhal, Dhameja, Panda (2018), S. 191.
[6] Vgl. hierzu und zum Folgenden Nakamoto (2008), S. 1.
[7] Vgl. hierzu und zum Folgenden Bianco (2023).
[8] Vgl. Chainlist.org (2023).

2.5 Smart Contracts

Smart Contracts (im Sinne von Ethereum) sind Programme, die auf der Ethereum-Blockchain ausgeführt werden.[9] Sie bestehen aus einer Sammlung von Code (ihre Funktionen) und Daten (ihr Zustand), die - ebenso wie ein Benutzer-Wallet - an einer bestimmten Adresse auf der Ethereum-Blockchain (oder einer anderen EVM-Chain - S. 2.4) residieren. Smart Contracts sind eine Art von Ethereum-Konto, was bedeutet, dass sie ein Guthaben haben und Ziel von Transaktionen sein können. Sie werden jedoch nicht von einem Benutzer kontrolliert, sondern sind im Netzwerk bereitgestellt und laufen wie programmiert. Benutzerkonten können dann mit einem Smart Contract interagieren, indem sie Transaktionen einreichen, die eine auf dem Smart Contract definierte Funktion ausführen. Außerdem können Regeln / Bedingungen definiert werden, nach denen Code automatisch durchgeführt wird. Standardmäßig können Smart Contracts nicht gelöscht werden, und Interaktionen mit ihnen sind irreversibel.

2.6 Nodes

Teilnehmer des Netzwerks, die über die gesamte Blockchain verfügen und die Transaktionen und Blöcke validieren, werden Nodes genannt.[10] Diese Nodes formen das dezentralisierte Netzwerk, das die Blockchain betreibt. Diese sind es auch, die den Code der Smart Contracts und die Transaktionen ausführen. Außerdem validieren sie die Transaktionen und Blöcke, die von anderen Nodes erstellt wurden, und erstellen neue Blöcke, die sie dann an das Netzwerk senden. Dazu verwenden sie einen Konsensalgorithmus, der bestimmt, welche Blöcke gültig sind (s. 2.8). Sowohl für das Ausführen der Smart Contracts als auch für das Validieren der Blöcke erhalten die Nodes eine Belohnung in Form von ETH (oder einer anderen Kryptowährung, je nach Chain). Diese Belohnung wird Gas genannt.

2.7 Gas Fee

Gas Fee ist die Gebühr, die für die Ausführung von Smart Contracts bezahlt werden muss.[11] Sie wird in der Währung der jeweiligen Chain (bspw. ETH) bezahlt und ist abhängig von der Rechenkomlexität der konkreten Operationen, die bei einer Transaktionen ausgeführt werden („Base Fee"). Hinzu kommt eine „Priority Fee", um die Transaktion attraktiver für Validatoren zu machen und so schneller ausgeführt zu werden. So ändert sich die optimale „Priority Fee" je nach Auslastung des Netzwerks.

[9] Vgl. hierzu und zum Folgenden Ethereum.org (2023f).
[10] Vgl. hierzu und zum Folgenden Bitcoin.org (2023); Ethereum.org (2023l).
[11] Vgl. hierzu und zum Folgenden Ethereum.org (2023e).

2.8 Konsensalgorithmus

Um sicherzustellen, dass bestehende Blöcke nicht verändert werden können und der Inhalt neuer Blöcke valide ist, wird ein Konsensalgorithmus verwendet.[12] Die bekanntesten Algorithmen sind:

- **Proof of Work (PoW):** Miner (Nodes) konkurrieren miteinerander, um ein mathematisches Problem zu lösen.[13] Der Erste, dem dies gelingt, erhält die Blockbelohnung und kann den den Block erstellen. Um einen Block zu erstellen, wird dementsprechend Zeit benötigt. Ein Angreifer müsste also mehr als die Hälfte der Rechenleistung des Netzwerks besitzen - um so die Blöcke schneller als alle anderen erstellen zu können -, um die Blockchain zu manipulieren.

- **Proof of Stake (PoS):** Es wird zufällig eine Node ausgewählt, die den nächsten Block erstellen darf und so die Belohnung erhält.[14] Es muss vorab eine Sicherheitsleistung (Stake) hinterlegt werden, die verloren geht, wenn die Node versucht, die Blockchain zu manipulieren. Je höher der Stake, desto höher die Wahrscheinlichkeit, dass die Node ausgewählt wird. So wird verhindert, dass ein Angreifer die Blockchain manipuliert, da er mehr als die Hälfte des Stakes besitzen müsste, um die Blockchain zu manipulieren.

- **Proof of Authority (PoA):** Es wird eine Liste von Nodes festgelegt, die die Blockchain validieren dürfen.[15] Dieser Algorithmus wird häufig bei privaten Chains verwendet, da den Nodes vertraut werden muss. So kann die Blockchain nicht manipuliert werden, ohne dass eine der Nodes dies zulässt. Dieser Algorithmus ist sehr schnell, da keine Rechenleistung benötigt wird oder eine Auswahl getroffen werden muss, um einen Block zu erstellen.

3 Eignung im Zahlungsverkehr

3.1 Ist-Zustand

Aktuell läuft die Abwicklung des Zahlungsverkehrs der Sparkassen über einen Großrechner (Mainframe) von IBM. [16] Für den Großrechner werden sogenannte „Jobs" - also Kurzprogramme, die eine bestimmte Aufgabe erfüllen - geschrieben. Dafür wird die Programmiersprache COBOL verwendet, welche hoch performant ist und so die Massendatenverarbeitung ermöglicht. Allerdings sind COBOL-Jobs an den

[12] Vgl. Nakamoto (2008), S. 2 f.
[13] Vgl. hierzu und zum Folgenden Nakamoto (2008), S. 3.
[14] Vgl. hierzu und zum Folgenden McKinsey & Company (2023), S. 2.
[15] Vgl. hierzu und zum Folgenden BinanceAcademy (2023).
[16] Vgl. hierzu und zum Folgenden Interviewprotokoll 1: Christian Krauthoff, S. A2.

Großrechner gebunden und immer weniger Entwickler sind in der Lage, diese zu schreiben. Daher werden zusätzlich Java-Jobs geschrieben, die ebenfalls auf dem Großrechner laufen, aber auch auf anderen Systemen ausgeführt werden könnten. Es ist das Ziel, vor allem die Verwendung von COBOL-Jobs - aber auch die Verwendung vom Großrechner im Allgemeinen - zu reduzieren und so auf andere Systeme zu migrieren. Neuentwicklung haben die klare Richtlinie in Java implementiert zu werden und nur kritische Themen - im Sinne von Zeit und Datenmasse - auf dem Großrechner zu belassen. [17] Es können teilweise 2 Millionen Transaktionen in ca. 20 Sekunden durchgeführt werden oder mehrere Jobs parallel laufen. Für weniger kritische Themen werden OSPE-Services verwendet, die auf einem anderen System laufen. Grundsätzlich laufen Prozesse des Zahlungsverkehrs wie folgt ab:

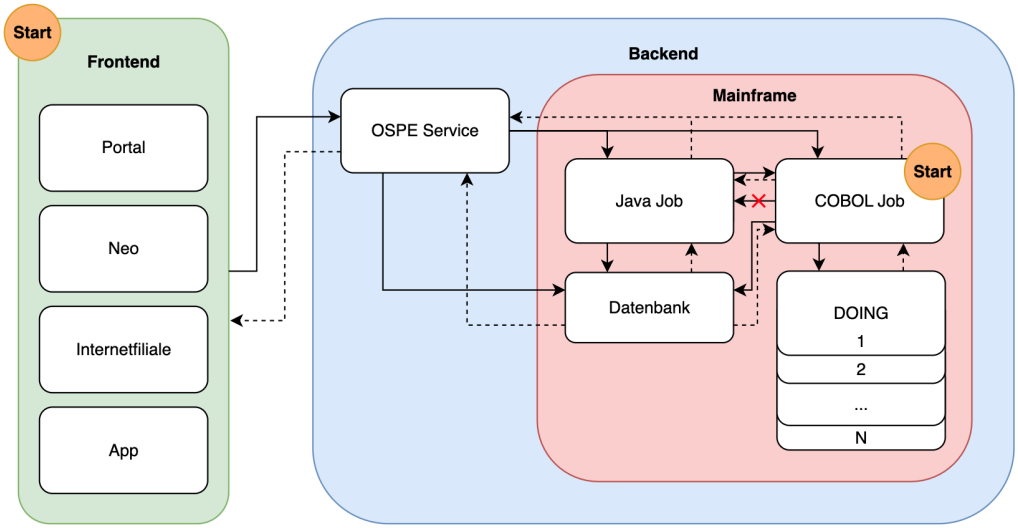

Abbildung 1: OSPlus-Diagramm

- Ein Mitarbeiter oder ein Endkunde setzt einen Prozess im Frontend in Gang (bspw. Überweisung oder Abfrage eines Kontostands). Dies kann über das Portal oder die Neo-Anwendung für Mitarbeiter oder das Online-Banking / die App für Endkunden geschehen.

  Ein Prozess kann ebenfalls durch einen COBOL-Job in Gang gesetzt werden (bpsw. bei Terminüberweisungen).

- Die Anfrage wird von einem OSPE-Service verarbeitet. Entweder kann dieser die Anfrage direkt (in in Verbindung mit anderen OSPE-Services) bearbeiten oder er ruft einen Java- oder COBOL-Job auf.

- Der Java-Job kann ebenfalls COBOL-Jobs aufrufen, andersherum ist dies nicht möglich.

[17] Vgl. Persönliches Gespräch mit Nils Pudenz - Entwickler AZV & ZV-Rekla (OE-4293); Interviewprotokoll 1: Christian Krauthoff, S. A2.

- Bei bestimmten Prozessen (bpsw. Transaktionen) müssen eine Anzahl an Prüfungen - die DOINGs - durchgeführt werden. (Z. B., ob die IBAN existiert, ob der Kontoinhaber der richtige ist, ob das Konto gedeckt ist oder auch, ob der Empfänger auf einer Sperrliste steht.)
- Ist diese Prüfung erfolreich, kann der Prozess fortgesetzt werden.
- Sowohl die OSPE-Services als auch die Java- und COBOL-Jobs greifen auf die Datenbank zu, um die benötigten Daten zu erhalten bzw. zu speichern.

Kostentechnisch ist es schwer, die Kosten pro Transaktion zu bestimmen, da es sich hierbei um Volumentverträge handelt. Den Sparkassen werden 1,00 € bis 1,50 € pro tausend Zahlungsein- oder -ausgängen berechnet.[18]

Um Daten nachvollziebar zu speichern - also so, dass auch nach Änderung oder eigentlicher Löschung noch der vorherige Stand ermittelbar ist -, werden Daten nie wirklich aktualisiert oder gelöscht, sondern es wird ein neuer Datensatz angelegt, der den aktuellen Stand der Daten enthält. Der alte Stand wird nur „deaktiviert", also mit einem Ablaufdatum versehen, sodass dieser nicht mehr verwendet wird.[19] Erst nach einer in der DSGVO festgelegten Zeit - im Falle von Kontodaten sind dies 10 Jahre - kann dieser Datensatz dann wirklich gelöscht werden.

3.2 DLS als Datenbank

Ein Use Case dieser Technologie im Bereich Zahlungsverkehr ist die Verwendung als reine Datenbank. Eine Blockchain erfüllt automatisch durch ihre Architektur die Fähigkeit Daten nachvollziehbar zu speichern. So können Transaktionen - also in diesem Fall Einträge in diese Datenbank - nicht rückgängig, nicht verändert und so nicht manipuliert werden.
Um ein DLS als Datenbank zu verwenden, gibt es einige Herausforderungen, die im Folgenden erläutert werden. Außerdem stellt sich die Frage, ob eine öffentliche oder private Blockchain verwendet werden sollte.

3.2.1 Öffentliche vs. private Chain

Bei der Verwendung einer Blockchain gibt es die Möglichkeit, einer öffentlichen Chain „beizutreten" oder dafür eine private Chain zu betreiben.[20] Eine öffentliche Chain ist für alle Teilnehmer offen und kann von jedem verwendet werden.[21] Eine privatee Chain hingegen ist nur für ausgewählte Teilnehmer zugänglich und wird i. d.

[18] Vgl. interne Preisliste OSPlus 2024.
[19] Vgl. hierzu und zum Folgenden A1.
[20] Vgl. Ethereum.org (2023g).
[21] Vgl. hierzu und zum Folgenden Ethereum.org (2023d).

R. von einer oder mehreren Organisationen / Unternehmen betrieben. Die Auswahl der Art der Chain kann an den Punkten Sicherheit / Unveränderlichkeit, Leistung, Kosten, Berechtigungen und Datenschutz / Anonymität erfolgen:

- **Sicherheit / Unveränderlichkeit:** Die Sicherheit und Unveränderlichkeit einer Blockchain wird durch ihren Konsensalgorithmus bestimmt. Eine öffentliche Chain wird durch die Interaktion von Tausenden unabhängigen Nodes gesichert, die von Einzelpersonen und Organisationen auf der ganzen Welt betrieben werden. Private Chains haben typischerweise eine kleine Anzahl von Nodes, die von einer oder wenigen Organisationen kontrolliert werden. Diese Nodes können streng kontrolliert, aber nur wenige müssen kompromittiert werden, um die Chain umzuschreiben oder betrügerische Transaktionen durchzuführen.

- **Leistung:** Bei privaten Chains können Hochleistungsnodes mit besonderer Hardware sowie ein anderer Konsensalgorithmus verwendet werden, um einen höheren Tansaktionsdurchsatz auf der Basisschicht (Layer 1) zu erreichen. Bei einer öffentlichen Chain kann ein hoher Durchsatz mit Hilfe von Layer 2 Skalierungslösungen erreicht werden.

- **Kosten:** Die Kosten für den Betrieb einer privaten Chain spiegeln sich hauptsächlich in der Arbeit wider, die Chain einzurichten und zu verwalten und die Server zu betreiben, auf denen sie läuft. Während es keine Kosten gibt, um sich mit dem Ethereum Mainnet zu verbinden, muss die Gas Fee (s. 2.7) für jede Transaktion in Ether bezahlt werden. Abhilfe können Transaction Relayers (s. 3.2.5.2) schaffen, sodass ein Endkunde diese Gebühr nicht selbst tragen muss.

  Einige Analysen haben gezeigt, dass die Gesamtkosten für den Betrieb einer Anwendung auf einer öffentlichen Chain niedriger sein können als beim Betrieb einer privaten Chain.[22]

- **Berechtigungen:** Bei privaten Chains können nur autorisierte Teilnehmer Nodes einrichten und Transaktionen durchführen. Bei öffentlichen Chains kann jeder Node einrichten und Transaktionen durchführen. So kann ebenfalls jeder auf jeden Contract zugreifen, also dessen gespeicherte Daten auslesen und Funktionen aufrufen. Daher müssen erstellte Contracts so implementiert werden, dass sie nur von den gewünschten Teilnehmern verwendet werden können.

- **Datenschutz / Anonymität:** Der Zugang zu Daten, die auf privaten Chains festgehalten wurden, kann frei vom Betreiber kontrolliert werden. Alle Daten, die auf einer öffentlichen Chain geschrieben wurden, sind für jeden einsehbar, so dass sensible Informationen off-chain gespeichert und übertragen oder verschlüsselt werden sollten. Es bestehen Designpattern, die dies erleichtern, sowie

[22] Vgl. EYBlockchain (2019), S. 14.

Layer 2 Lösungen, die Daten abtrennen und von Layer 1 fernhalten können (s. 3.2.5.1).

Ebenso sind alle Transaktionen auf einer öffentlichen Chain öffentlich einsehbar, sodass die Anonymität der Teilnehmer nicht gewährleistet werden kann (s. 3.2.4.1).

Da sich der Scope dieser Arbeit in der FI aufhält und so das Betreiben einer privaten Chain nicht mehr dezentral sondern zentral wäre, da sie dann nur von der FI betrieben werden würde, wird im weiteren Verlauf nur auf die Verwendung einer öffentlichen Chain eingegangen (s. weiterführend 5). Grundsätzlich hätte das Betreiben einer privaten Chain viele Vorteile, da so die Transaktionskosten gespart werden könnten und die Daten nicht öffentlich einsehbar wären. Allerdings würde sich so das System kaum von dem heutigen unterscheiden.

3.2.2 Datenspeicherung

Alle auf der Chain gespeicherten Daten sind öffentlich zugänglich. So können sensible Daten, die sich nicht häufig ändern - wie bspw. Namen oder Kontonummern - verschlüsselt oder weiterhin zentral gespeichert werden. Außerdem muss die Verbindung der Konten zur Walletadresse gespeichert werden. So können sich andere Daten hinter der Walletadresse verbergen und nicht direkt mit dem eigentlichen Kontobesitzer in Verbindung gebracht werden (s. weiterführend 3.2.4). Um den Kontostand eines Kunden widerzuspiegeln, gibt es die Möglichkeit diesen als Layer 1 Währung oder als Token darzustellen.
Bei der Darstellung des Kontostandes als Layer 1 Währung (z. B. ETH) müsste jeder Kunde ein Wallet erhalten / besitzen, das den Wert des Kontos in einer Layer 1 Währung enthält. Das Problem dabei ist, dass diese nicht den Euro darstellt. So müsste der Wert immer in eine andere Währung umgerechnet werden, was zu zusätzlichen Kosten führt. Außerdem ist der Wert einer Layer 1 Währung sehr volatil, was zu Problemen führen kann, wenn der Wert des Kontos nicht mit dem Wert der Layer 1 Währung übereinstimmt. Diese Darstellungsart kommt also nicht in Frage.

3.2.2.1 ERC20 Token

Es gibt diverse Standards für Smart Contracts, die bestimmte Funktionen und Eigenschaften definieren. Einer dieser Standards ist der ERC20 Token Standard. Dieser definiert die Schnittstellen eines Smart Contracts, der als Token verwendet werden soll.[23] Ein Token kann dabei eine beliebige Repräsentation eines Vermögenswertes sein. In diesem Smart Contract wird die Anzahl der Token gespeichert, die eine bestimme Adresse (also Benutzer-Wallet oder Smart Contract) besitzt. Außerdem werden Funktionen definiert, um u. a. Token von einer Adresse zu einer anderen zu

[23] Vgl. hierzu und zum Folgenden Ethereum.org (2023c).

versenden, die Anzahl der Token einer Adresse abzufragen und anderen Adressen die Erlaubnis zu erteilen, Token von der eigenen Adresse zu versenden.[24]

Es gibt bestimmte ERC20 Token, die den Wert anderer Assets (unter anderem auch den Euro) abbilden. Diese werden Stablecoin genannt.[25] Das Problem daran - sowie auch bei der Layer 1 Währung - ist, dass diese einen tatsächlichen Wert haben und so die Bank dieses Geld nicht für eigene Geschäfte verwenden kann.

Es wäre also sinnvoll, einen eigenen Token zu erstellen, um den Wert eines Kontos darzustellen, ohne dass dieser einen tatsächlichen Wert hat. So kann die Bank diesen Wert für eigene Geschäfte verwenden, ohne dass der Kunde dadurch einen Verlust erleidet. Außerdem kann so gewährleistet werden, dass nur Kunden der Bank diesen Token verwenden können, da die Bank die einzige ist, die diesen Token ausgibt. Außerdem muss der erstellte Token nicht zwingend die genauen Schnittstellen eines ERC20 Tokens, sondern lediglich die Anforderungen der Bank erfüllen.

3.2.3 Datenverfügbarkeit

Um bei komplexen oder größeren Anfragen nicht die gesamte Blockchain durchsuchen zu müssen bzw. bestimmmte Abfragen überhaupt möglich zu machen, werden die Daten zusätzlich in einer Datenbank indexiert und gespeichert.[26] Sobald eine Transaktion auf der Chain ausgeführt wird, die für die Datenbank relevant ist, wird ein Event ausgelöst, der die Daten in die Datenbank schreibt bzw. bestehende Daten aktualisiert. Diese Datenbank kann zentral oder ebenfalls dezentralisiert sein.

3.2.4 Anonymität

In einem öffentlichen Blockchain-Netzwerk sind alle Transaktionen öffentlich einsehbar. Anonymität wird dadurch gewährleistet, dass die Identität eines Teilnehmers von der Walletadresse getrennt ist.[27] Allerdings können mehrere Transaktionen eines Wallets miteinander in Verbindung gebracht werden, und so die Anonymität eines Teilnehmers gefährden. So wird - zumindest bei Bitcoin - dazu geraten, jede Adresse nur genau zweimal zu verwenden. Einmal, um Bitcoin zu empfangen und einmal, um dieses wieder zu versenden.[28] Dieses Problem würde ebenfalls bestehen, wenn die Kontostände in einem öffentlichen Netzwerk abgebildet werden würden.

[24] Vgl. OpenZeppelin (2023); Ethereum.org (2023c).
[25] Vgl. hierzu und weiterführend Presidents's Working Group on Financial Markets, Federal Deposit Insurance Corporation, Office of the Comptroller of the Currency (2021), S. 4.
[26] Vgl. hierzu und im Folgenden TheGraph (2023); Moralis (2022a).
[27] Vgl. Nakamoto (2008), S. 6.
[28] Vgl. BitcoinDeveloper (o. J.)

3.2.4.1 Account-based vs. UTXO-based Chain

Bei einer UTXO-based Chain (bspw. Bitcoin) werden die Anzahl an Coins pro offener Transaktion gespeichert.[29] Eine Adresse besitzt also genauso viele Coins wie die Summe der offenen Transaktionen, die an diese Adresse gesendet wurden. Eine Transaktion kann mehrere Ein- und Ausgänge haben, wobei die Summe gleich sein muss. So können auch zwei offene Transaktionen unterschiedlicher Adressen zusammengefasst und einem Empfänger zugesendet werden.

Bei einer Account-based Chain (bspw. EVM-Chains) werden die Anzahl an Coins pro Account / Adresse gespeichert.[30] So wird bei einer Transaktion die Anzahl an Coins von einem Account auf einen anderen Account übertragen, sprich, die Anzahl an Coins wird von einem Account abgezogen und dem anderen Account hinzugefügt. Die Art von Chain wird vor allem bei Smart Contract fokussierten Blockchains verwendet. Um in diesem Modell Coins mehrerer Adressen zusammenzufassen, sind mehrere Transaktionen notwendig.

3.2.4.2 HD-Wallets

Bei einem HD-Wallet werden beliebig viele Keys (sprich Adressen) aus einem Hauptschlüssel (Seed) generiert.[31] So kann ein Wallet verwendet werden, dabei aber für jede Transaktion eine neue Adresse verwendet und so eine hohe Anonymität gewährleistet werden.[32] Diese Art von Wallet sind der heutige Standard für UTXO-based Chains. Allerdings können HD-Wallets nicht für Account-based Chains genutzt werden, da diese keine Keys verwenden, sondern die Anzahl an Coins pro Account speichern.

3.2.4.3 UTXO-based Token

Aufbauend auf der Idee einen neuen Token zu kreieren (s. 3.2.2.1) könnte dieser auch als UTXO-basierter Token implementiert werden. So werden nicht wie üblich bei einem ERC20 Token der Coinstand pro Adresse gespeichert, sondern die offenen Transaktionen pro Adresse. Bei einem Transfer können dann beliebig viele offene Transaktionen als In- und Output verwendet werden. Dabei könnte die Methode so implementiert werden, dass der Contract selbst die Transaktionen zusammensucht, die in Summe den gewünschten Betrag ergeben oder es werden beim Aufruf die zu verwendenden Transaktionen als Parameter übergeben. Letzteres würde weniger Aufrufe benötigen und so Transaktionskosten (Gas) sparen. Auf diese Weise kann ein HD-Wallet fähiger Token erstellt werden, der auf einer Account-based Chain (also auch einer EVM-Chain) verwendet werden kann.

---

[29] Vgl. hierzu und zum Folgenden sowie weiterführend Singhal, Dhameja, Panda (2018), S. 182 ff.

[30] Vgl. hierzu und zum Folgenden Crypto APIs Team (2022).

[31] Vgl. hierzu und weiterführend Binnewies (2023), S. 8 ff.

[32] Vgl. Singhal, Dhameja, Panda (2018), S. 231.

3.2.5 Transaktionskosten und -durchsatz

Ethereum kann nur eine geringe Anzahl an Transaktionen pro Sekunde (ca. 20TPS bis 30TPS) verarbeiten.[33] Daher liegen die Transaktionskosten je nach Netzwerkauslastung zwischen ca. $2 bis $20.[34] Um die Skalierbarkeit zu erhöhen und Transaktionskosten zu senken, werden Layer 2 Lösungen eingesetzt.[35]

3.2.5.1 Layer 2 Lösungen

Layer 2 ist ein grundsätzlicher Begriff für Lösungen, die „off-chain" - also außerhalb der eigentlichen Blockchain (Layer 1) - betrieben werden, dort die Transaktionen gebündelt und nur die Ergebnisse auf die Layer 1 Chain geschrieben werden.[36] Dabei gibt es verschiedene Ansätze, unter anderem die Folgenden:

- **Sidechains:** Eine Sidechain ist eine eigene Blockchain, die mit der Layer 1 Chain verbunden ist und so parallel zu dieser läuft. Diese Chain hat ihre eigenen Regeln zum Thema Konsensalgorithmus und Blöcken.[37]

  Da hier eine eigene Blockchain verwendet wird, ist diese Lösung sehr flexibel und kann für viele verschiedene Anwendungsfälle verwendet werden. Allerdings wird nicht die Sicherheit der Layer 1 Chain übernommen, sondern es muss der Sidechain vertraut werden, da ein eigener Konsensalgorithmus verwendet wird.[38]

- **State Channels:** State Channels ermöglicht es, vielfache Transaktionen „off-chain" zu tätigen und dabei nur zwei Transaktionen auf die Layer 1 Chain zu schreiben - das Öffnen und Schließen des Channels. Es gibt zwei Arten von Channels: Payment Channels und State Channels.[39]

  Payment Channels sind sicher, da sie den Konsensalgorithmus der Layer 1 Chain verwenden, können allerdings nur für spezifische Anwendungszwecke verwendet werden (bspw. Coin- oder Token-Transaktionen).

  State Channels können für beliebige Transaktionen verwendet werden, sind allerdings unsicherer, da die Validität der Transaktionen nur durch die Teilnehmer des Channels geprüft werden und nicht durch die Layer 1 Chain.

- **Rollups:** Rollups führen die Transaktionen ebenfalls „off-chain" aus. Die Ergebnisse werden dann in einem Layer 1 Block gesammelt und so in die Layer 1 Chain geschrieben.[40] So werden Rollups mit der Layer 1 Chain gesichert. Es gibt zwei Arten von Rollups: Optimistic Rollups und Zero Knowledge Rollups.

[33] Vgl. Ronis (2023).
[34] Vgl. Etherscan (2024).
[35] Vgl. Das (2024).
[36] Vgl. Ethereum.org (2023h).
[37] Vgl. hierzu und weiterführend Ethereum.org (2023i).
[38] Vgl. Moreland (2023).
[39] Vgl. hierzu und zum Folgenden sowie weiterführend Ethereum.org (2023j).
[40] Vgl. hierzu und zum Folgenden Ethereum.org (2023h).

Optimistic Rollups gehen davon aus, dass Off-Chain-Transaktionen gültig sind und veröffentlichen keine Nachweise für die Gültigkeit der auf der Chain veröffentlichten Transaktionsbatches.[41] Nachdem ein Rollup-Batch auf Ethereum eingereicht wurde, gibt es ein Zeitfenster (die sogenannte Challenge-Periode), in dem jeder die Ergebnisse einer Rollup-Transaktion durch Berechnung eines Betrugsnachweises anfechten kann („Fraud Proof"). Wenn der Betrugsnachweis erfolgreich ist, wird die Transaktion neu ausgeführt und der Zustand des Rollups entsprechend aktualisiert.

In ZK-Rollups wird ein Stapel (engl. „Batch") von Transaktionen auf dem Ethereum-Netzwerk auf Korrektheit überprüft.[42] Nach dieser Prüfung wird der Stapel von Transaktionen als endgültig betrachtet, genau wie jede andere Ethereum-Transaktion. Dafür wird ein kryptographischer Gültigkeitsnachweise (allgemein als Zero-Knowledge-Nachweise bezeichnet) verwendet. Bei jedem Stapel von Off-Chain-Transaktionen generiert der ZK-Rollup-Betreiber einen Gültigkeitsnachweis für diesen Stapel. Sobald der Nachweis generiert ist, wird er an Ethereum übermittelt, um den Roll-up-Stapel in der Layer 1 Chain einzubinden.

Um das hohe Datenvolumen der Sparkassen zu bewältigen, müssen Layer 2 Lösungen verwendet werden. Wobei auch hier fraglich ist, ob diese der Masse an Transaktionen standhalten können (es wird mit ca. 100.000 TPS gerechnet).[43] Preislich soll eine Transaktion weitaus weniger als $0,1 kosten.[44]

3.2.5.2 Transaction Relayers

Um es dem Account zu ermöglichen Transaktionen zu tätigen, ohne ETH zu besitzen, besteht die Möglichkeit, dass ein Dritter die Transaktion für den Account tätigt und so auch bezahlt.[45] Dieser Dritte wird Transaction Relayer genannt. Dabei wird die Transaktion vom Benutzer zwar signiert, dann aber nicht direkt ans Netzwerk gesendet und ausgeführt, sondern an den Relayer übersendet, der die Transaktion dann ausführt. Es muss auf der Chain ein Smart Contract existieren, der die Transaktionen entgegennimmt, die Signatur prüft und dann die Transaktion durchführt.[46] Allerding können nur speziell dafür angepasste Funktionen eines Smart Contracts auf diese Weise ausgeführt werden, da der eigentliche „Ausführende" nicht der Sender ist und deshalb anders geprüft werden muss (nicht über *msg.sender* sondern über bspw. *msgSender()*).[47]

[41] Vgl. hierzu und zum Folgenden sowie weiterführend Alchemy (2022).
[42] Vgl. hierzu und zum Folgenden sowie weiterführend ZkSync (2024).
[43] Vgl. Ethereum.org (2023b).
[44] Vgl. Polygon (2023).
[45] Vgl. hierzu und zum Folgenden sowie weiterführend Sandford et al. (2020); Bloemen, Logvinov, Evans (2017).
[46] Vgl. Clement (2020).
[47] Vgl. Lundfall (2020); Sandford et al. (2020).

Transaction Relayers wären sinnvoll, wenn ein Kunde selbst bpsw. Überweisungen o. ä. über das Netzwerk durchführen kann. Da allerdings nur die FI selbst Transaktionen durchführt - ein Endkunde würde dann durch die Services der Sparkassen und so der FI das System verwenden - und diese selbst die Kosten für die Transaktionen tragen muss, ist es nicht notwendig Transaction Relayers zu verwenden.

3.3 DLS als Mainframersatz

Um den Mainframe wirklich ablösen zu können, reicht es nicht aus lediglich die Datenbank zu ersetzen. Es muss auch die Verarbeitung und Prüfung der Daten übernommen werden. Kurzum müssen also die COBOL- und Java-Jobs sowie die DOINGs, die akutell auf dem Mainframe laufen, ersetzt werden.
Eine Möglichkeit wäre es, all diese Jobs durch Smart Contracts zu ersetzen. Allerdings wären so alle Verarbeitungen und Prüfungen öffentlich einsehbar, was sensible Daten (wie bspw. Name oder IBAN) offenlegen würde. Außerdem wäre dies durch die Transaktionskosten sehr teuer.
Da durch das hohe Datenaufkommen der Sparkassen die Verarbeitung der Daten sehr performant sein muss, ist es ebenfalls nicht möglich, die Verarbeitung der Daten auf der Blockchain selbst durchzuführen. Wie bereits beschrieben muss also eine Layer 2 Lösung verwendet werden (s. 3.2.5.1). Da die FI die einzige Instanz ist, die Transaktionen durchführt und so keine gesonderte Prüfung der Transaktionen notwendig ist, bietet sich hier ein (modifiziertes) Optimistic Rollup an. (Die Challange-Periode ist nicht erforderlich, es wird also immer der durch das Rollups kreierte Batch verwendet.) Die FI wickelt also den Zahlungsverkehr weiterhin auf eigenen Systemen ab und übersendet nur die Ergebnisse an die Layer 1 Chain. So werden die Prüfungen und Verarbeitungen der Daten auch nicht öffentlich ausgeführt und sind nicht einsehbar.
Bestimmte Finanzprodukte lassen sich dennoch gut durch Smart Contracts abbilden und so automatisieren. Darunter zählen z. B. automatische Zinsvergaben am Monats- oder Jahresabschluss oder auch Termin- oder Daueraufträge.

3.3.1 Optimistic Rollups in Verbindung mit Cloud Computing

Eine mögliche Alternative den Mainframe abzulösen, ist die Verwendung von Cloud Computing. Cloud Computing wird nach dem National Institute of Standards and Technology durch die folgenden fünf Eigenschaften definiert:[48]

- **On-Demand Self-Service:** Der Nutzer kann die benötigten Ressourcen (bspw. Rechenleistung, Speicherplatz, etc.) selbstständig und ohne menschliche Interaktion mit dem Anbieter bereitstellen.

[48] Vgl. hierzu und zum Folgenden Baun et al. (2011), S. 5.

- **Broad Network Access:** Die Ressourcen sind über das Netzwerk verfügbar und können von verschiedenen Plattformen (bspw. Desktop, Laptop, Smartphone, etc.) abgerufen werden.
- **Resource Pooling:** Die Ressourcen des Anbieters werden gebündelt und können so von mehreren Nutzern verwendet werden.
- **Rapid Elasticity:** Die Ressourcen können schnell und dynamisch an die Bedürfnisse des Nutzers angepasst werden.
- **Measured Service:** Die Nutzung der Ressourcen wird quantitativ und qualitativ überwacht, so dass eine nutzungsabhängige Abrechnung sowie eine Validierung der Dienstqualität möglich ist.

Der größte Vorteil für die FI wäre hier die gute Skalierbarkeit der Ressourcen und die nutzungsabhängige Abrechnung.

Ein größer Nachteil dieser Technologie ist die Datensicherheit, da sich diese in der Hand des Anbieters befindet und so nicht kontrolliert werden kann.[49] Es besteht generell keine Kontrolle über die Art und Weise der Speicherung. So ist ein Wechsel des Anbieters sehr aufwendig, da die Daten möglicherweise nicht einfach migriert werden können.[50]
Hier können DLS Abhilfe schaffen. So können die Optimistic Rollups auf einem Cloud System ausgeführt, die Daten aber dennoch in der Blockchain gespeichert werden. So sind sie tranzparent und nach eigenen Vorstellungen gespeichert, aber die Verarbeitung der Daten kann auf einem Cloud System erfolgen. Darüber hinaus können die Daten nicht nur vom verwendeten Anbieter selbst aus eingesehen werden - da sie sich im öffentlcihen Netzwerk befinden - und eine Migration ist so einfacher möglich. Man ist also nicht stark an einen Anbieter gebunden.

3.3.2 Erweiterbarkeit

Sollten Finanzprodukte in Smart Contracts umgesetzt werden oder sich die Funktionsweise der Optimistic Rollups in Zukunft ändern, ist es wichtig, dass diese Funktionen erweiterbar / weiterentwickelbar sind. Die Grundidee einer Blockchain basiert allerdings darauf, dass einmal geschriebene Daten - und so auch Smart Contract - nicht mehr verändert werden können (sie sind „immutable").[51] Es gibt drei Möglichkeiten, wie Smart Contracts erweitert werden können:[52]

- **Contract Migration:** Bei der Contract Migration löst ein neuer Contract den alten ab. Alle Daten, die im alten Contract gespeichert wurden, müssen

[49] Vgl. hierzu und zum Folgenden Baciu (2015), S. 99.
[50] Vgl. Padamkar (2023).
[51] Vgl. Moralis (2022b).
[52] Vgl. hierzu und zum Folgenden sowie weiterführend Ethereum.org (2023k).

also in den neuen Contract übertragen werden. Die Datamigration ist meist sehr aufwendig und mit hohen Gas-Kosten verbunden.

- **Data Seperation:** Bei der Data Seperation werden Daten und Logik in zwei verschiedene Contracts aufgeteilt. So kann der „Logik-Contract" ersetzt werden, ohne dass die Daten migriert werden müssen. Hier müssen aufwendige Sicherheitsmaßnahmen getroffen werden, um zu gewährleisten, dass nur der Logik-Contract die Daten verändern kann.

- **Proxy:** Bei der Verwendung eines Proxys wird ein zusätzlicher Contract verwendet, der als Schnittstelle zu dem eigentlichen Contract dient. Es ist möglich, dass ein Contract die Funktionen eines anderen Contracts in seinem Kontext - also mit seinem Speicher - aufruft. Der Proxy Contract hält also den Speicher, während der dahintergeschaltete Contract die Logik enthält. So kann der Proxy Contract einfach ausgetauscht werden, ohne dass die Daten migriert werden müssen.

  Bei Proxys gibt es mehrere Architekturen, die sich in der Art und Weise, wie der Proxy mit dem dahintergeschalteten Contract interagiert, unterscheiden. Häufig kommt zusätzlich ein Admin Contract zum Einsatz, der die Berechtigung hat, den Logik-Contract zu ändern.

Da mehrere Finanzprodukte in Smart Contract abgebildet werden könnten, bietet es sich an für jedes Produkt einen eigenen Contract zu verwenden und diese dann über einen gemeinsamen Proxy anzusprechen.

4 Schlussfolgerung

Die Verwendung von DLS also reine Datenbank verfehlt den Zweck, den Großrechner abzulösen. Hinzu kommt, dass die Verwendung als Datenbank auch keine sonderlich gute Alternative ist, da dies mit einem sehr hohem Aufwand und Komplexität verbunden ist und die Verwendung teurer und langsamer ist. Außerdem besteht auch kein Problem mit der aktuellen Art und Weise wie Daten gespeichert werden, dass mit DLS gelöst werden würde.
Die Verwendung von Optimistic Rollups als Ersatz für den Großrechner ist schon eher eine Alternative, allerdings auch sehr komplex. Die Verwendung von DLS in Verbindung mit Cloud Computing könnte allerdings eines der größten Probleme, die mit Cloud Computing bestehen - die Datensicherheit -, lösen. Allerdings ist es fraglich, ob diese Technologie die Performance des Großrechners erreichen kann. Dafür müsste diese Idee prototypisch umgesetzt und getestet werden, was aufgrund der hohen Komplexität und des benötigten Testdatenumfangs mit einem sehr hohen Aufwand verbunden wäre.

Daher ist die Verwendung von DLS für den Zahlungsverkehr der Sparkassen eher wenig geeignet. Wenn, dann nur in Verwendung mit Cloud Systemen.

5 Ausblick

Die grundlegende Idee eines DLS ist es, dessen Teilnehmer zu vernetzen, ohne zentrale Instanzen zu verwenden und so kein Vertrauen in diese setzen zu müssen (s. 2.3). Da in dieser Arbeit die Verwendung von DLS lediglich im Scope einer einzelnen Instanz - der FI - betrachtet wurde, und daher keine Vernetzung stattfindet, fällt dieser Vorteil vollständig weg. Eine weitaus sinnvollere Verwendung im Berech Zahlungsverkehr wäre die Vernetzung mehrerer Banken, um so bpsw. Transaktionen auszuführen. Dafür kann auch ein privates Netzwerk geschaffen werden, da dies dann auch dezentral wäre und so die Vorteile eines DLS genutzt werden können. Die Bundesbank hat hierzu einen Bericht geschrieben, in dem sie sich positiv für diese Technologie im Auslandszahlungsvekehr äußert, um diesen so zu vereinfachen und zu beschleunigen.[53]

[53] Vgl. Deutsche Bundesbank (2017), S. 44.

6 Quellenverzeichnis

Alchemy (2022):
How Do Optimistic Rollups Work (The Complete Guide), https://www.alchemy.com/overviews/optimistic-rollups, Stand: 18.01.2024.

Baciu, E. (2015):
Advantages and Disadvantages of Cloud Computing Services, from the Employee's Point of View; in: National Strategies Observer, 2. Jg., Heft 1, S. 95-101.

Baun, C. / Kunze, M. / Nimis, J. / Tai, S. (2011):
Cloud Computing - Web-basierte dynamische IT-Services, 2. Aufl., Springer.

Bianco, A. (2023):
What are EVM Chains?, https://www.datawallet.com/crypto/evm-chains, Stand: 21.12.2023.

BinanceAcademy (2023):
Proof of Authority Explained, https://academy.binance.com/en/articles/proof-of-authority-explained, Stand: 08.01.2024.

Binnewies, T. (2023):
Analyse des Aufbaus eines Crypto-Wallets sowie die Einbindung dessen in eine Banking-App am Beispiel der Sparkassen-Banking-App der Finanz Informatik.

Bitcoin.org (2023):
What Is A Full Node?, https://bitcoin.org/en/full-node#what-is-a-full-node, Stand: 08.01.2024.

BitcoinDeveloper (o. J.):
Transactions, https://developer.bitcoin.org/devguide/transactions.html, Stand: 12.01.2024.

Bloemen, R. / Logvinov, L. / Evans, J. (2017):
EIP-712: Typed structured data hashing and signing - A procedure for hashing and signing of typed structured data as opposed to just bytestrings., https://eips.ethereum.org/EIPS/eip-712, Stand: 18.01.2024.

Chainlist.org (2023):
Chainlist - Helping users to connect to EVM powered networks, https://chainlist.org, Stand: 22.12.2023.

Clement (2020):
ETH - Pre-Paid smart contract so users don't pay transactions, https://ethereum.stackexchange.com/questions/88268/eth-pre-paid-smart-contract-so-users-dont-pay-transactions/88276#88276, Stand: 18.01.2024.

Crypto APIs Team (2022):
UTXO and Account-Based Blockchains, https://cryptoapis.io/blog/7-utxo-and-account-based-blockchains, Stand: 12.01.2024.

Das, L. (2024):
What Are Ethereum Layer 2 Blockchains and How Do They Work?, https://www.ledger.com/academy/topics/blockchain/what-are-ethereum-layer-2-blockchains-and-how-do-they-work, Stand: 17.01.2024.

Deutsche Bundesbank (2017):
Distributed- Ledger- Technologien im Zahlungsverkehr und in der Wertpapierabwicklung: Potenziale und Risiken; in: Monatsbericht, September 2017, S. 35-50.

Ethereum.org (2023a):
Blocks, https://ethereum.org/en/developers/docs/blocks/, Stand: 03.01.2024.

Ethereum.org (2023b):
Danksharding, https://ethereum.org/roadmap/danksharding, Stand: 28.01.2024.

Ethereum.org (2023c):
ERC-20 Token Standart, https://ethereum.org/en/developers/docs/standards/tokens/erc-20/, Stand: 22.12.2023.

Ethereum.org (2023d):
Ethereum Mainnet for Enterprise, https://ethereum.org/en/enterprise/, Stand: 08.01.2024.

Ethereum.org (2023e):
Gas and Fees, https://ethereum.org/developers/docs/gas, Stand: 28.01.2024.

Ethereum.org (2023f):
Introduction to Smart Contracts, https://ethereum.org/en/developers/docs/smart-contracts/, Stand: 21.12.2023.

Ethereum.org (2023g):
Private Ethereum for Enterprise, https://ethereum.org/en/enterprise/private-ethereum/, Stand: 08.01.2024.

Ethereum.org (2023h):
Scaling, https://ethereum.org/developers/docs/scaling, Stand: 17.01.2024.

Ethereum.org (2023i):
Sidechains, https://ethereum.org/developers/docs/scaling/sidechains, Stand: 17.01.2024.

Ethereum.org (2023j):
State Channels, https://ethereum.org/developers/docs/scaling/state-channels, Stand: 17.01.2024.

Ethereum.org (2023k):
Upgrading Smart Contracts, https://ethereum.org/en/developers/docs/smart-contracts/upgrading, Stand: 27.01.2024.

Ethereum.org (2023l):
What are nodes and clients?, https://ethereum.org/en/developers/docs/nodes-and-clients/, Stand: 08.01.2024.

Etherscan (2024):
Average Transaction Fee Chart, https://etherscan.io/chart/avg-txfee-usd, Stand: 29.01.2024.

EYBlockchain (2019):
Total cost of ownership for blockchain solutions.

Greeh, A. / Camilleri, A. (2017):
Blockchain in Education.

LedgerAcademy (2023):
Distributed Ledger Meaning, https://www.ledger.com/academy/glossary/distributed-ledger, Stand: 21.12.2023.

Lundfall, M. (2020):
ERC-2612: Permit Extension for EIP-20 Signed Approvals - EIP-20 approvals via EIP-712 secp256k1 signatures, https://eips.ethereum.org/EIPS/eip-2612, Stand: 19.01.2024.

Majaski, C. (2023):
Distributed Ledgers: Definition, How They're Used, and Potential, https://www.investopedia.com/terms/d/distributed-ledgers.asp, Stand: 21.12.2023.

McKinsey & Company (2023):
What is proof of stake?.

Moralis (2022a):
Create a Moralis Dapp (Desprecated), https://v1docs.moralis.io/moralis-dapp/getting-started/create-a-moralis-dapp, Stand: 28.01.2024.

Moralis (2022b):
What are Upgradable Smart Contracts? Full Guide, https://moralis.io/what-are-upgradable-smart-contracts-full-guide/, Stand: 27.01.2024.

Moreland, K. (2023):
What is Plygon (MATIC)?, https://www.ledger.com/academy/blockchain/what-is-polygon-matic, Stand: 17.01.2024.

Nakamoto, S. (2008):
Bitcoin: A Peer-to-Peer Electronic Cash System.

OpenZeppelin (2023):
ERC20.sol, https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol, Stand: 22.12.2023.

Padamkar, P. (2023):
Advantages and Disadvantages of Cloud Computing, https://www.educba.com/advantages-and-disadvantages-of-cloud-computing/, Stand: 27.01.2024.

Polygon (2023):
Polygon PoS, https://polygon.technology/polygon-pos, Stand: 29.01.2024.

Presidents's Working Group on Financial Markets / Federal Deposit Insurance Corporation / Office of the Comptroller of the Currency (2021):
Report on Stablecoins.

Ronis, J. (2023):
Understanding Ethereum's Layer 1 and Layer 2: Differences, Adoption, and Drawbacks, https://www.wilsoncenter.org/article/understanding-ethereums-layer-1-and-layer-2-differences-adoption-and-drawbacks, Stand: 29.01.2024.

Sandford, R. / Siri, L. / Tirosh, D. / Weiss, Y. / Forshtat, A. / Croubois, H. / Tomar, S. / McCorry, P. / Venturo, N. / Vogelsteller, F. / John, G. (2020):
ERC-2771: Secure Protocol for Native Meta Transactions - A contract interface for receiving meta transactions through a trusted forwarder, https://eips.ethereum.org/EIPS/eip-2771, Stand: 18.01.2024.

Singhal, B. / Dhameja, G. / Panda, P. S. (2018):
Beginning Blockchain, 1. Aufl., Apress.

Tamalio, M. (2023):
Wie funktioniert die Distributed-Ledger-Technologie?, https://btv-bank.de/wissen/distributed-ledger-technologie/, Stand: 21.12.2023.

TheGraph (2023):
About the Graph, https://thegraph.com/docs/en/about/, Stand: 28.01.2024.

ZkSync (2024):

Intro to Rollups, https://docs.zksync.io/build/developer-reference/rollups.html#what-are-zk-rollups, Stand: 18.01.2024.

IV Anhangsverzeichnis

V Anhang

Anhang 1 - Interviewprotokoll 1: Christian Krauthoff

Protokoll:	Gespräch mit Christian Krauthoff
Teilnehmer:	Christian Krauthoff - Abteilungsleiter AZV & ZV-Rekla (OE-4293)
	Tobias Binnewies - Dualer Student
Thema:	Produktionsdatenspeicherung / Transaktionskosten / Großrechner
Dauer:	30 min

Frage: Wie wird sichergestellt, dass Produktionsdaten (bspw. Kontodaten) nicht manipuliert werden können?
Antwort: Als Mitarbeiter braucht man ein Produktionsrecht und ein recht sensitive Daten einsehen zu dürfen, um auf die Daten mit einem Auftragsgrund zugreifen zu können. Der Auftragsgrund besteht dann durch ein Ticket, also quasi eine „Beschwerde" durch ein Institut. Nur dann kann ein FI-Mitarbeiter auf Produktionsdaten zugreifen. Um Daten abzuändern muss noch ein weiteres Recht beantragt werden, das nur wenige Mitarbeiter haben. Außerdem kommt das Vieraugenprinzip zum Einsatz, sodass immer mindestens zwei Mitarbeiter mit diesem Recht das SQL-Statement einsehen müssen, bevor es ausgeführt wird.
Source Code wird maschinell geprüft, also bspw. dass keine festen IBANs, Namen oder ähnliches im Code fest abgefragt werden. Außerdem wird immer ein Code Review von „Unabhängigen„ - also Entwicklern, die an diesem Projekt nicht beteiligt waren - durchgeführt, um so die Qualität des Codes zu gewährleisten.

Frage: Wie werden Daten bei Änderung nachvollziehbar gespeichert?
Antwort: Wenn Daten „gelöscht" werden oder auch geändert werden, werden diese nie wirklich gelöscht sondern nur deaktiviert. So gibt es bei jeder gespeicherten Zeile in der Datenbank ein Feld „Gültig bis", welches erst einmal leer ist. Ändert sich nun etwas an der Zeile, wird das Feld „Gültig bis" mit dem Datum der Änderung gefüllt und eine neue Zeile mit den neuen Daten wird angelegt. So ist eine Löschung oder Änderung jederzeit nachvollziehbar.
In der DSGVO ist festgelegt, sind Datenlöschungfristen festgelegt. Kontodaten haben eine Löschfrist von 10 Jahren, d. h. diese müssen 10 Jahre lang gespeichert werden und können dann erst wirklich gelöscht werden.

Frage: Wie hoch sind die Kosten pro Transaktion?
Antwort: Die Informationen dazu stehen im Produktkatalog. Außerdem gibt es SQLs, die einem die genauen Kosten für das jeweilige Insitut auswerfen.
Die Kosten für die Finanz Informatik sind pro Transaktion schwer anzugeben, da es sich hierbei um Volumentverträge handelt. Hierbei wird eine Lastspitze der Großrechners im gesamten Jahr festgelegt und demnach abgerechnet.

Frage: Gibt es Projekte um vom Großrechner wegzukommen?
Antwort: Ja, man möchte klar vom Großrechner wegkommen - und auch vorallem von IBM abhängig zu sein. Es gibt das Projekt „COBOL to Java", also von COBOL / IBM weg und so nur noch die kritischen Themen dort zu halten, also im Rahmen von Zeit und Massendatenverarbeitung und den Rest alles auslagern in Java. Allerdings laufen die Java-Jobs zum Teil auch auf dem Großrechner, das Zeil ist jedoch - sofern möglich - die Großrechnerjobs immer weiter zu reduzieren.
Das Problem ist immer die Geschwindigkeit und Massendatenverarbeitung. Neue Systeme sind immer daran gescheitert. Bspw. schafft der Großrechner, wenn ein Job mit ca. 1,4 Millionen Transaktionen anstartet ca. 20 Sekunden. Oder auch wenn z. B. die DKB am Monatsende mit ihren ca. 1,8 Millionen Konten Zinsrechnung, Abschußschreibung oder auch Umbuchung durchführt, laufen mehrere Jobs parallel. Das ist mit anderen Systemen schwer abbildbar, bzw. auch schnwer einmal zu testen. Cloud Systeme werden hier getestet, aber einen richtigen Harttest mit bspw. 2 Millionen Jobs parallel gab es nicht.

Frage: Was sind Geschäftsprozesse?
Antwort: Geschäftsprozesse sind quasi ein „Fahrplan", dort steht drin, was bspw. von der Erfassung bis zur Buchung mit den Daten passiert, also ob z. B. ob aus 5€ noch 15€ werden, da es noch Gebühren gibt oder ist das nicht erlaubt. Es wird genau geregelt, welcher Prozess wann, wie und in welcher Reihenfolge abläuft. So kann genau nachgewiesen werden, was genau mit den Daten passiert ist. Es gibt dann „Datenflussdiagramme„ für einen genauen Prozess um bspw. bei Beschwerde genau nachvollziehen zu könnne, warum was genau passiert ist. Es ist technisch nicht möglich einen Prozess zu umgehen, da die Daten immer durch die Prozesse laufen. Außerdem können bpsw. keine Daten gelöscht oder abgeändert werden, wenn dies nicht im Geschäftsprozess vorgesehen ist.