PANTERCON©

START YOUR IDEA WITH THE PANTER

Aubergine Paper

Social Media



Inhalt Tech Paper Hydra Blockchain

1. Ebenen der Hydra Hybrid Blockchain

1.1 Allgemeines

Um hohe Performance und geringe Speicherkapazität zu gewährleisten besteht die Hydra aus mehreren Ebenen. Ähnlich dem Plasma Ansatz wird hier ebenfalls auf MapReduce-Funktionen gesetzt und durch die Parallelisierung der Verifizierungsprozesse in mehrere erzwungene Childchains mit Merkle Trees durchgeführt.

Allein durch die Verwendung von zugeordneten Sidechains (Childchains) zu einer Hauptchain erhöht sich die Skalierbarkeit enorm.

1.2 Mainchain

Stellt die klassische Hauptvariante und Herzstück des Blockchain–Systems dar. Bei der Hydra besteht diese Mainchain aus vielen verschiedenen Ebenen, die nachfolgend erklärt werden.

1.3 Sidechains

Die Sidechains sind eigene separate Blockchains, die einem "Two-Way-Peg" an ihrer übergeordneten Blockchain befestigt werden. Dieser "Two-Way-Peg" (Zerberus) regelt die Austauschbarkeit von Objekten zwischen der übergeordneten Block- und den Sidechains. Hierfür wird eine fixe Rate definiert. Es ist auch möglich, an eine Sidechain mehrere Sidechains zu hängen.

Somit können auch gesamte Hierarchien abgebildet werden.

Um diese hierarchische Abbildung zu benennen wird immer von einer "Parent"-Chain und der untergeordneten "Child"-Chain gesprochen.

Der technische Ablauf ist wie folgt: Benutzer müssen immer in der übergeordneten Kette Objekte an eine gewisse Ausgabeadresse senden. Dort erfolgt die Sperrung der Objekte, sodass diese in dieser Chain nicht mehr verwendet werden können, damit es durch die Benutzer zu keiner Doppelnutzung kommen kann. Nach der Verifikation dieser Transaktion werden die Objekte in der Sidechain freigegeben. Der Benutzer kann diese jetzt dort verwenden. Der umgekehrte Fall tritt ein, wenn man von einer Sidechain zur Hauptkette zurückkehrt.

1.4 Parallelisierung von Verifizierungsprozessen.

Übersteigt die aktuelle Anzahl von Transaktionen zur Verifizierung einen vorgegebenen Wert, wird automatisch ein neuer Verifizierungsprozesse parallel gestartet.

Dieser Vorgang kann sich mehrmals wiederholen und wird auf eine gewisse Anzahl an parallelen Prozessen limitiert sein. Abhängig vom verwendeten Consensus Alghorithmus und der Anzahl der Verifizierungspunkte kann es hier zu starken Abweichung der Performance

im direkten Vergleich kommen.

Die Anzahl der Transaktionen, die einen neuen Parallelprozess auslösen und das Limit der Parallelprozesse, werden im Zuge der Umsetzung durch Tests ermittelt, um hier ein Optimum zu ermitteln.

Technisch erfolgt die Umsetzung mit Merkle-Trees und in erzwungen Sidechains.

Somit sollte anfänglich eine Transaktionsrate von 50.000 pro Sekunde erreichbar sein.

In der letzten Ausbaustufe von Hydra wird ein Vielfaches davon realisierbar sein.

1.5 ECHIDNA (Stammdaten der Hydra Objekte)

Beinhaltet die Stammdaten aller Ebenen aller Child-, Sidechains und der Mainchain. In ECHIDNA werden nur die Stammdaten der Chains und ihrer Objekte verifiziert hinterlegt. Hier finden keine klassischen Transaktionen statt. Diese Ebene wird ausschließlich für die Funktionsfähigkeit aller anderen Chains verwendet. Die gesamte Gebührenverwaltung wird ebenfalls in ECHIDNA verwaltet.

Prinzipiell ist es Möglichkeit, dass in Sidechains andere Gebühren oder zusätzliche Gebühren erhoben werden, die dann einer zugewiesenen Organisation überwiesen werden. Dies hängt vom Consensus Alghorithmus ab und ob die Sidechains „public“ oder „private“ sind.

Um eine Vereinfachung für den Enduser zu erhalten werden Gebühren der Transaktion in der Mainchain in 3 Prioritätsstufen (hoch, mittel, niedrig) eingeteilt. Priorität hoch ist mit hohen Gebühren verbunden, garantiert aber eine sofortige Überweisung usw.

In der finalen Ausbaustufe wird es Benutzern auch möglich sein, selbst neue Gebühren zu erstellen.

1.6 NYX (anonyme Transaktionen)

Der Teilbereich NYX soll anonyme Transaktion ermöglichen.

Hierzu gibt es jedoch noch rechtlichen Klärungsbedarf.

1.7 PLUTOS (B2B Transaktionen)

Hier werden die B2B Transaktionen gespeichert. Ähnlich wie bei anderen public Chains sind diese Adressen theoretisch einsehbar. Durch die Benutzerberechtigung wird es möglich sein, dass die Eigentümer einer Wallet zeitgesteuerte Rechte an eine andere Stelle übergeben. Bsp.: Ein Unternehmen gewährt Leserechte eines Kontos für die Buchhalterin des Steuerberaters mit einer zeitlichen Einschränkung (Bsp.: nur die letzten 3 Monate).

1.8 HERMES (HYDRA Objekt Marktplatz)

Hier werden die Transaktionen von Warenobjekt dokumentiert. In diesem Bereich liegt der Fokus einerseits

auf Nachvollziehbarkeit im Warenverkehr und anderseits auf Datenbereitstellung. Schwerpunkte wird hier besonders auf die nachfolgenden Punkte gelegt:

- Seriennummern
- Chargen
- Logistiktokens

Im Bereich der Datenbereitstellung geht es darum, die Daten für B2B Prozesse für andere User bereitzustellen. Mögliche Anwendungsfälle hierfür sind, dass Lieferanten Artikel mit allen Attributen (Spezifikation von Maßen, Textbeschreibungen, Verfügbarkeiten und auch Preisen bereitstellen). Über die Rechtevergabe ist es möglich diese nur vordefinierten Partner zur Verfügung zu stellen. Diese wiederum können dann mit diesen Daten arbeiten und ihre ERP Systeme befüllen.

1.9 DEMETER (Eigentumsverhältnisse Rechte als Handelsgut)

Dieser Bereich bietet ein sehr hohes Potential an Anwendungsfällen. Mit dem Übergang der Eigentumsverhältnisse an einen oder mehrere andere User werden automatisch auch Rechte mitvergeben. Diese können endgültig sein oder zeitlichen Einschränkungen unterliegen.

1.10 HERA (Rechte für Chains, Smart Contracts)

1.10.1 Rechte allgemein

In diesem Bereich werden die Rechte der einzelnen Benutzer, Ihre Rollen und damit erhalten Berechtigungen auf den jeweiligen Chains dokumentiert. Hierzu wird das Berechtigungssystem noch ausführlich besprochen. Die wichtigsten Bereiche hier sind, dass an die Rechte zwei Arten von Zeitsteuerung gebunden sind.

- Das Recht selber gilt nur für einen bestimmten Zeitraum (bsp.: Urlaubsvertretung, dann kann ein Recht für einen Mitarbeiter für 14 Tage erstellt werden)
- Das Recht unterliegt einer zeitlichen Begrenzung (Bsp.: Mitarbeiter darf nur die Transaktionen des aktuellen Quartals einsehen)

1.10.2 Zuweisung der Rechte

Die Zuweisung der Rechte erfolgt nach einem zweistufigen Prinzip. Die Rechte können erstellt werden und sind automatisch gespeichert. Diese werden jedoch erst durch die Verifizierung aktiv und die Gebühr fällt an.

1.11 HADES (Archiv)

Der Bereich HADES dient der Archivierung von Daten. Über die Phönix Methode werden diese von der Main-

chain in die HADESChain verlagert. Diese Funktion kann für Sidechain in der finalen Ausbaustufe ebenfalls verwendet werden. Die Phönix Methode wird nachfolgend noch detaillierter erklärt.

1.12 ARCHIMEDES (Hydra Objekte)

Hier werden die Details zu den Hydra Objekten gespeichert. Allgemein wird der Begriff Objekt für alle möglichen Arten von Tokens und neu generierten Dinge verwendet.

Des Weiteren ist es auch möglich, gänzlich neue Arten von Objekten zu generieren, die mit den "klassischen" Tokens nicht vergleichbar sind. Hier ist das Limit die Fantasie.

- Bsp.: ToDoListen fürs Projektmanagement,
- Verwaltung von Grundstücksgrenzen,
- Kalorienzähler für Weightwatchers
- Wandernadelverwaltung usw.

1.13 Testnetzwerk

Das Testnetzwerk verfügt über alle Eigenschaften des produktiven Systems. Die Unterschiede sind hier, dass die Bewegungsdaten (Transaktionen) jedes Quartal gelöscht werden, nicht aber die Smart Contracts. Zusätzlich wird über KRONOS die Möglichkeit geschaffen, die Zeit zu simulieren. Somit können zeitgesteuerte Verträge als Beispiel mit 10-facher Geschwindigkeit getestet werden. Dies würde bedeuten, dass ein Ver-

trag der real 30 Tage läuft, in 3 Tagen durchgelaufen wäre. KRONOS besitzt nicht nur die Fähigkeit, die Zeit zu beschleunigen, sondern auch an gewisse Punkte zu stoppen oder bewusst an gewisse Zeitpunkte zu gehen, um von diesem Zeitpunkt aus zu testen.

1.14 Tokens

Es wird eine Menge an bereits vordefinierten Tokens geben, die bereits über gewisse Eigenschaften verfügen. Solche klassische Tokens werden Security Token, Utility Token, Membership Token oder Access Token. Bei jedem Token wird es aber auch möglich sein, frei Funktionen dazu zu programmieren.

2. Rollenbasierendes zeitgesteuertes Benutzerberechtigungskonzept

2.1 Allgemein

Dieses Konzept kombiniert die klassische rollenbasierende Benutzerberechtigung mit einer zeitgesteuerten Rechtevergabe und einem zeitgesteuerten Zugriff auf Bewegungsdaten.

2.2 Bestandteile Konzept

Es ist möglich, einem User eine oder mehrere Rollen (Admin, Sachbearbeiter usw.) zu zuweisen. Diesen Rollen sind mit speziellen Rechten. Ein Benutzer darf eine Aktion ausführen, wenn er aufgrund einer (oder

mehrerer) seiner Rollen das Recht zugewiesen bekommen hat. Ein Benutzer darf eine Aktion ausführen, wenn er aufgrund einer (oder mehrerer) seiner Rollen das Recht zugewiesen bekommen hat. Hier wird es möglich sein, dass diese Rollen eine zeitliche Begrenzung haben. Bsp.: Übertragen einer Rolle auf einen anderen Mitarbeiter wegen Urlaubsvertretung.

In der finalen Ausbaustufe wird es möglich sein, Rechte auch noch auf Benutzerebene unabhängig von Rollen weiter zu individualisieren.

2.2.1 Organisationen

Organisationen können am besten als juristische Personen, Firmen, NGO usw. erklärt werden.

Diese können Sidechains betreiben oder auch mit der Mainchain arbeiten. Für die von Ihnen erstellten Objekte (Tokens, Sidechains usw.) haben sie die Möglichkeit, Berechtigungen zu vergeben.

2.2.2 Organisationseinheiten

Diese sind die Einheiten, aus denen eine Organisation besteht und bildet wie im klassischen Organigramm die Abteilungen ab. Diese Organisationseinheiten können dann Rollen zugewiesen werden, die sich dann auf die Benutzer vererben.

2.2.3 Benutzer Login

Generell muss sich ein User im Mainnet einloggen, um sich dann in die anderen Ebenen bewegen zu können. Beim Einloggen werden die Berechtigungen der verschiedenen Ebenen geladen und ermöglichen die Navigation in den einzelnen Bereichen.

2.2.4 Rechte

Als Rechte wird die Freigabe oder Verhinderung einer Aktion verstanden. Bsp.: Lesen von Transaktionsdaten, Erstellen von Objekten.

2.2.5 Rollen

Als Rollen werden Funktionen von Personen verstanden, die notwendig für ihre Ausübung über mehrere Rechte verfügen müssen. Diese Rollen können zum Bsp.: Administrator, Sachbearbeiter usw. sein.

2.2.6 Rollen Rechte

Diesen Rollen werden dann Berechtigungen zugewiesen. Als Beispiel darf ein Sachbearbeiter Verkauf und Transaktion des letzten Quartals lesen, jedoch keine Überweisungen tätigen.

2.2.7 Benutzer Rollen

Hier werden den einzelnen Benutzern eine oder mehrere Rollen zugewiesen. Dies bedeutet, dass es theoretisch möglich sein kann, dass sich Rollenrechte wi-

dersprechen. Hier greift dann das „günstigste“ Recht. Beispiel ein User ist der Rolle „Sachbearbeiter Verkauf“ zugeordnet und darf deswegen keine Überweisungen tätigen. Er ist aber auch der Rolle „Geschäftsführung“ zugeteilt. Diese Rolle hat sehr wohl die Berechtigung Überweisungen zu tätigen, somit wird dieses Recht gewährt.

2.3 Zeiteinschränkung

2.3.1 Temporäre Rechte

Wie eingangs bereits erwähnt, gibt es die Möglichkeit, Rechte temporär zu vergeben. Diese können für einen gewissen Zeitraum oder mit einem Ablaufdatum versehen werden. In der finalen Version wird es möglich sein, über Smart Contracts diese ereignisgesteuert durchführen zu lassen. Bsp.: Nach Erhalt einer Zahlung wird ein Recht um einen Monat verlängert.

2.3.2 Zugriff auf Daten mit begrenztem Zeitraum

Ein wichtiger Punkt ist ebenfalls die Möglichkeit, Bewegungsdaten mit Begrenzungen abrufbar zu machen. Dies kann ein statischer Zeitraum oder ein dynamischer Zeitraum sein.

So wäre es denkbar, dass gewisse User nur die Transaktionen seit Jahresbeginn sehen dürfen und andere immer nur alle Transaktion, die nicht älter sind als 3 Monate.

3. Consensus Algorithmus

In der Mainchain wird ein dem Proof of Importance ähnlicher Algorithmus verwendet werden. Jedem einzelnen Punkt wird ein Wichtigkeitswert zugewiesen, der sich aus verschiedenen Faktoren wie Anzahl der PANX, Haltedauer, Höhe und Anzahl von Transaktionen usw. zusammensetzt. Diese Methode hilft sicherzustellen, dass alle Rechner im Netzwerk miteinander übereinstimmen. Benutzer mit hoher „Wichtigkeit“ können „ernten“ und Belohnungen verdienen.

Bei Hydra können konkret durch dieses Verfahren HXP geerntet werden. Diese wiederum werden einerseits benötigt, um Aktionen auszuführen, oder können ganz einfach verkauft werden.

Für selbst erstellt Hydraobjekte und vor allem in den Sidechains können abweichende consensus Algorithmen verwendet werden.

In der finalen Ausbaustufe werden hierfür die gängigsten Consensus Algorithmen bereitgestellt und es wird möglich sein, eigene Consensus Verfahren zu programmieren.

4. HERAKLES

Herakles ist die GUI für das Erstellen neuer Side- und Childchain, die Rechteverwaltung und die gesamte Objektverwaltung. Dies bildete den Kernteil für das gesamte Verwaltungssystem.

Besonderes Augenmerk wird auf eine intuitive Bedienbarkeit gelegt. Des Weiteren wird es hier Schnittstellen für das Interagieren mit den einzelnen Bereichen der Blockchain oder einzelnen Smartcontracts geben.

5. Phönix

Phönix ermöglicht das Verbrennen von Tokens und ihre gleichzeitige Wiedergeburt nicht nur innerhalb einer Chain. Diese Funktionsweise kann auch für customize Token eingesetzt werden oder wird in den Hydra Main Chains verwendet. Am einfachsten ist der Vergleich mit einer Schluss- und Eröffnungsbilanz. Hierbei werden für alle bebuchten Konten die Salden erstellt und in die neue Buchungsperiode übernommen. Das passiert meist zum Jahresende kann aber bei Hydra in frei wählbaren Zeiträumen passieren.

Hier erfolgt eine Art Saldierung ins Phönix Protokoll. Die Mainchain wird zu einer Hades Archivchain und eine neue Mainchain wird erstellt. Aus dem Phönix Protokoll werden jetzt die saldierten Werte in die neue Mainchain übertragen. Die Transaktionen sind dadurch nicht verloren gegangen, sondern können über die Hades ArchivChain weiter eingesehen werden.

Für den User wird es so sein, dass es so wirkt, als wären diese Daten nach wie vor in der Main Chain.

Durch diese Methodik wird die Blockchain immer wieder klein gehalten und es können Archive erstellt wer-

den. Theoretisch wäre es auch möglich, so ein Archiv zu vernichten.

Da diese Methodik auch in Sidechains zur Verfügung steht, kann es hier abhängig von der Funktionsweise und den Daten die gespeichert werden, auch Berührungspunkte mit der DSGVO geben. Um diesen gesetzlichen Aspekten Genüge zu tun, kann hier die Phönix Methode verwendet werden.

6. Really Smart Contracts

6.1 Allgemein

Die Programmiersprache zur Erstellung von Really Smart Contracts wird C# sein. Ziel ist es, Verträge so „intelligent" gestalten zu können, dass diese automatisiert laufen und alle wichtigen Bestandteile eines herkömmlichen Vertrags abdecken.

Wie bei diesen klassischen Verträgen können hierfür Laufzeiten, Kündigungsfristen, Zeitpunkte usw. definiert werden.

Dennoch wird es abhängig von Vertragstypen entweder eine „Abort" oder – „Rollback"–Funktion geben. Die „Abort"-Funktion bricht den Vertrag ab und setzt ihn außer Kraft bsp.: Dauerauftrag für das Gehalt eines Mitarbeiters, wenn der Mitarbeiter gekündigt hat. Die „Rollback"–Funktion kehrt einen Vertrag um. Bsp.: Ein ICO hat das Softcap nicht erreicht und die Investoren erhalten ihre Einlage wieder zurück.

In den Contracts kann für die „Abort“ oder „Rollback“-Funktion eine eigene Prozesskette definiert werden Bsp.: Lieferant bricht den Vertrag ab und muss dafür eine Pönale in vordefinierten Höhe an den Kunden zahlen. Ideale Anwendungsfälle hierfür sind Mengen- und Abrufcontrakte. Weitere Anwendungsgebiete könnten Provisionszahlungen mit externen Unternehmen, Abos mit Endkunde sowie auch Konsignationsverträge sein.

Die Verträge müssen zwingend über eine Laufzeit verfügen.

Die Aktivierung eines Vertrags obliegt dem Auftragsersteller, jedoch können die Aufträge mit Berechtigung versehen werden, die das Zustimmen der anderen Vertragspartner erzwingen.

6.2 Automatismen

Bei den Contracts wird besonders Augenmerk auf Automatismen gelegt. Hierfür werden zwei Kernelemente berücksichtigt: ereignisgesteuerte und zeitgesteuerte Ausführung von Funktionen. Als Ereignis kann der Erhalt einer Transaktion verstanden werden, das dann eine gesamte Prozesskette auslöst. Zeitgesteuert wäre, wenn von einer Filiale die Beträge jeden Tag automatisch um Mitternacht in die Zentrale überwiesen werden.

6.3 Standardisierung durch Modularisierung von Smart Contracts

Wir stellen standardisierte Module bereit, die durch den User miteinander kombiniert werden können. Jedes Modul liefert Input- und Output- Parameter mit und verfügt über die Fähigkeiten, in Schleifen ausgeführt zu werden. Als Beispiel kann hier ein Modul „zeitgesteuerte Preisfindung“ genannt werden.

So wäre es möglich, hier eine Tabelle zu hinterlegen, die zu jedem Tag einen neuen Preis ausgibt.

Durch die Kombination von Modulen können hier ereignisgesteuerte Prozessketten ebenfalls abgebildet werden.

6.4 Customizing von Modulen

Dennoch wird es möglich sein, für spezielle Anwendung statt dem Modul Code selbst einen Code zu schreiben und zu hinterlegen. Bei jedem Modul wird es auch möglich, statt dem vorgeschlagenen Code einen benutzerdefinierten Code einzugeben. Somit könnte hier das Modul zeitgesteuerte Preisfindung so umprogrammiert werden, dass der Preis nach den ersten 100 Verkäufen am Tag automatisch um 10% steigt. Hierzu müsste noch ein neuer Input Parameter für die Anzahl der täglichen Verkäufe hinterlegt werden.

Generell wird ein selbstgeschriebener Code in eine Datenbank zwischengespeichert. Erst durch das Akti-

vieren des Smart Contracts wird der Code in die Blockchain geschrieben und erhält seine Gültigkeit.

6.5 Interaktion mit Contracts während der Laufzeit

Vor dem Aktivieren des Contracts müssen Parameter, die Interaktion zulassen, definiert werden und mit Berechtigungen versehen werden. Als Beispiel kann hier eine Vertragsverlängerung genannt werden. Abhängig von der Berechtigung kann hier der Vertragseigentümer dies allein durchsetzen oder es muss von den anderen Vertragspartnern eine Zustimmung eingeholt werden. Generell wird es eine GUI für das Interagieren mit Smart Contracts geben. Dort können abhängig von Berechtigungen die User, die vom Smart Contract Eigentümer bereitgestellten Funktionen abrufen. Außerdem wird es eine Schnittstelle zur Stapelverarbeitung geben. So können dann in einem Contract Daten durch den Contract Owner upgeloadet werden. Bsp.: Mehrere Adressen, die alle auf eine Whitelist gesetzt werden sollen, oder Auszahlungen an mehre Adressen mit verschiedenen Beträgen (Bsp.: Airdrop oder Bounty Programme).

6.6 Abbruch bei Nichterfüllung

Für den Fall der Nichterfüllung eines Vertrags werden Abbruchereignisse definiert. Bsp.: Zahlung wurde nicht durchgeführt. Dadurch wird für die benachteiligte Partei das „Exit“ Szenario freigegeben. In den meisten

Fällen ist das die „Abort“ Funktion, kann aber auch, wenn so definiert und vom Vertragstyp möglich, zu einer „Rollback“ Funktion führen. Die benachteiligte Vertragspartei muss diese Funktionen ausfüllen und manuell aktivieren.

7. Das “Zeitproblem”

Um eine einheitliche Zeit für die Blockerstellung zu gewährleisten (Erfassungszeit), wird generell unix time verwendet. Parallel wird pro Block die Zeit des Verifizierungspunkts gespeichert (Editor Zeit). Hier findet im Vorfeld eine Überprüfung mit einer zentralen Zeiteinheit statt. Die Editor Zeit wird auf Abweichung geprüft und sofern diese nicht mehr als 13 Sekunden beträgt, durchgeführt.

8. Merkle Tree / Merkle-Proof

Die Verifizierung erfolgt mit Merkle Trees.

Ein Merkle-Baum ist im weitesten Sinne eine Art von Zusammenstellung von Datenblöcken, die darauf beruht, die Blöcke in einer Art Baumstruktur darzustellen. Jeder Ast enthält nur wenige Blöcke, die zusammengefasst, gemeinsam gehasht werden und dann zu einem weiteren Ast führen. Jeder dieser Äste durchläuft jetzt den gleichen Prozess, der so oft wiederholt wird, bis die Gesamtzahl der verbleibenden Hashes nur noch ein Wert ist: der „Wurzelhash“.

Ein Merkle-Proof besteht aus dem „Wurzelhash“ des Baumes und dem des „Astes“, der aus allen Hashes besteht, die vom Ast zur Wurzel gehen.

Somit kann überprüft werden, ob das Hashing zumindest für diesen Ast sowohl konsequent bis zum Ende des Baumes durchgeführt wurde und ob die Position des Astes sich im Baum tatsächlich an dieser Stelle im Baum befindet.

Die Anwendung ist einfach: es gibt eine große Datenbank und der gesamte Inhalt der Datenbank wird in einem Merkle-Baum gespeichert, in dem die Wurzel des Merkle-Baums öffentlich bekannt und vertrauenswürdig ist (z.B. wurde er von genügend vertrauenswürdigen Parteien digital signiert, oder es gibt viele Beweise für die Arbeit daran). Ein Benutzer kann einen Merkle-Proof anfordern und nach Erhalt des Proofs überprüfen, ob er korrekt ist und ob der empfangene Wert tatsächlich an der überprüften Position in der Datenbank mit diesem bestimmten Root liegt.

In Zuge der Umsetzung wird nicht nur auf binäre Merkle-Trees gesetzt, sondern auf eine komplexere Variante ähnlich dem Patricia-Bäume Prinzip.