Technisches Informationsblatt

Harmony Team

Version 2.0

1. Einführung

Seit der Veröffentlichung des Bitcoin-Whitepapers (Informationsblatts) im Jahr 2008 hat sich das Konzept der Blockchain weltweit verbreitet. Während dezentrales Geld und Anwendungen zu gut publizierten Ideen werden, haben Konzept-Beschränkungen den Kerngedanken von Bitcoin in Frage gestellt. Die ursprüngliche Bitcoin-Blockchain wurde als Peer-to-Peer-Zahlungssystem konzipiert [13], das es Menschen ermöglicht, Werte ohne Vermittler wie Banken oder Zahlungsabwickler zu transferieren. Als Bitcoin jedoch an Popularität gewann, wurde sein Leistungs Engpass aufgrund seines begrenzten Durchsatzes von ~7 Transaktionen pro Sekunde (TPS) deutlich, und seine Kosten als Zahlungssystem wurden unerschwinglich.

Im Jahr 2014 plante Buterin et. al. [27] eine neue Blockchain-Infrastruktur namens Ethereum, die es Entwicklern ermöglichte, verschiedene Arten von Blockchain-Anwendungen mit "Smart Contracts" (schlauen Verträgen) zu erstellen. Ethereum hat das Skalierbarkeitsproblem jedoch nicht gelöst und mit seinen ~15 TPS keine Anwendungen mit hohem Transaktionsdurchsatz wie Spiele oder dezentrale Vermittlungen unterstützt.

Angesichts der Leistungseinschränkungen von Ethereum und Bitcoin schlugen viele Blockchain-Projekte verschiedene Lösungen vor [3,4,5,6,7,8,9,10,24,25], die versuchen, den Transaktionsdurchsatz zu erhöhen. Verschiedene Blockchains (Blockketten) [3,4,5,6,24,25] schlugen vor, den Proof-of-Work (PoW) Konsens durch den Proof-of-Stake (PoS) Konsens zu ersetzen. Andere Blockchains wie z.B. EOS verwenden Delegated Proof of Stake (DPoS), bei denen die Block-Antragsteller durch Abstimmung und nicht durch einen algorithmischen Prozess auf der Chain (Kette) gewählt werden. Projekte wie IOTA ersetzten die Datenstruktur der Blockchain durch eine DAG-Datenstruktur (Directed Acyclic Graph); die die Begrenzung der sequentiellen Verarbeitung von Transaktionen aufhebt.

Diese vorgeschlagenen Lösungen können jedoch keine signifikanten Leistungssteigerungen erzielen, ohne andere kritische Aspekte wie Sicherheit und Dezentralisierung zu opfern. Die Skalierbarkeitslösung, die sowohl Sicherheit als auch Dezentralisierung gewährleistet, ist Sharding (aufsplitten), das mehrere Gruppen (d.h. Shards) von Prüfern erstellt und sie Transaktionen gleichzeitig verarbeiten lässt. Dadurch steigt der gesamte Transaktionsdurchsatz linear mit zunehmender Anzahl der Shards. Zilliqa [12] war die erste öffentliche Blockchain, die vorschlug, das Skalierbarkeitsproblem beim Sharding anzugehen. Der Sharding-Ansatz von Zilliqa ist jedoch in zweierlei Hinsicht unzureichend. Erstens teilt es die Speicherung von Blockchain-Daten nicht auf (State Sharding). Dadurch wird verhindert, dass Maschinen mit begrenzten Ressourcen am Netzwerk teilnehmen und die Dezentralisierung eingeschränkt wird. Zweitens ist der Sharding-Prozess von Zilliqa anfällig für einen Übernahmefall mit nur einem Shard, da er auf PoW als Zufallsgenerator setzt.

Wir stellen Harmony vor, die nächste Generation einer sharding-basierten Blockchain, die vollständig skalierbar, nachweislich sicher und energieeffizient ist. Harmony geht auf die Probleme bestehender Blockchains ein, indem es die besten Forschungsergebnisse und technischen

Praktiken in einem optimal abgestimmten System kombiniert. Insbesondere macht Harmony Durchbrüche in folgenden Aspekten:

- **Voll skalierbar:** Harmony splittert nicht nur die Netzwerkkommunikation und Transaktionsbestätigung wie Zilliqa, sondern auch den Blockchain-Zustand. Das macht Harmony zu einer voll skalierbaren Blockchain.
- **Secure Sharding:** Der Sharding-Prozess von Harmony ist nachweislich sicher dank des DRG-Prozesses (Distributed Randomness Generation), der unvorhersehbar, unvoreingenommen, verifizierbar und skalierbar ist. Harmony verändert auch das Netzwerk auf eine nicht auffallende, unterbrechungsfreie Art und Weise, um langsam anpassungsfähigen Gegnern (Malware, Hacks etc.) vorzubeugen.
- **Effizienter und schneller Konsens:** Im Gegensatz zu anderen Sharding-basierten Blockchains, die PoW zur Auswahl von Validatoren benötigen, basiert Harmony auf PoS und ist damit energieeffizient. Konsens wird mit einem linear skalierbaren BFT-Algorithmus erreicht, der 100 mal schneller ist als PBFT.
- **Adaptive-Schwellenwert PoS:** Die Schwelle der Einsätze, die ein Knoten (Node) benötigt, um dem Netzwerk beizutreten, wird basierend auf dem Volumen der gesamten Einsätze so angepasst, dass böswillige Angriffe ihre Macht nicht auf einem einzigen Shard konzentrieren können. Außerdem ist die Schwelle niedrig genug, so dass auch PoS Teilnehmer mit nicht so vielen Coins am Netzwerk teilnehmen und Belohnungen verdienen können.
- **Skalierbare Netzwerkinfrastruktur:** Mit dem RaptorQ Quellcode kann Harmony Blöcke schnell innerhalb von Shards oder über das Netzwerk verbreiten, indem es den Adaptive Information Dispersal Algorithmus verwendet. Harmony verwendet auch das Kademlia-Routing [37], um Cross-Shard-Transaktionen zu erreichen, die logarithmisch mit der Anzahl der Shards skalieren.
- **Konsistente Cross-Shard-Transaktionen:** Harmony unterstützt Cross-Shard-Transaktionen mit Shards, die direkt miteinander kommunizieren. Ein atomarer Verriegelungsmechanismus wird verwendet, um die Konsistenz von Cross-Shard-Transaktionen sicherzustellen.

Durch Innovationen sowohl auf Protokoll- als auch auf Netzwerkebene bietet Harmony der Welt ein skalierbares und sicheres Blockchain-System, das in der Lage ist, die aufstrebende dezentrale Wirtschaft zu unterstützen. Harmony wird Anwendungen ermöglichen, die bisher auf Blockchain Ebene nicht möglich waren, darunter hochvolumige dezentrale Börsen, interaktive Fair Games (Spiele), Visa-Zahlungssysteme und Internet-of-Things-Transaktionen. Harmony ist bestrebt darin, Vertrauen für Milliarden von Menschen und eine radikal gerechte Wirtschaft für alle zu schaffen.

2. Konsens Mechanismus

Das Konsensus Protokoll ist eine Schlüsselkomponente jeder Blockchain (Blockkette). Es bestimmt, wie sicher und schnell Blockchain-Validierer (Prüfer) einen Konsens über den nächsten

Block erzielen. Das erste Blockchain-Konsensus Protokoll, das Bitcoin antreibt, ist der Proof-of-Work (PoW) Konsens. PoW ist ein Prozess, bei dem Miner (Schürfer) um die Lösung eines kryptographischen Rätsels ringen - der Gewinner erhält das Recht, den nächsten Block vorzuschlagen und erhält einige Token-Belohnungen. Die Sicherheits Annahme von PoW ist, dass mehr als 50% der Hashing-Power von ehrlichen Knoten (besser bekannt als “Nodes”) kontrolliert wird. Bei einer solchen Annahme ist die Regel für den Konsens, dass die längste Kette die ausschlaggebende sein wird, und deshalb wird der PoW-Konsens auch als *chain-based consensus* (kettenbasierter Konsens) bezeichnet.

Eine weitere Art von Konsens Protokoll, das seit mehr als zwei Jahrzehnten in der Wissenschaft erforscht wird, heißt PBFT (Practical Byzantine Fault Tolerance) [14]. In PBFT wird ein Knoten (Node) als "Leiter" gewählt, während die restlichen Knoten (Nodes) "Prüfer" sind. Jede Runde des PBFT-Konsenses umfasst zwei Hauptphasen: die Vorbereitungsphase und die Commit-Phase. In der Vorbereitungsphase sendet der “Leiter” seinen Vorschlag an alle “Prüfer”, die ihrerseits ihre Stimmen über den Vorschlag an alle anderen senden. Der Grund für die erneute Übertragung an alle Prüfer ist, dass die Stimmen jedes Prüfers von allen anderen Prüfern gezählt werden müssen. Die Vorbereitungsphase endet, wenn mehr als $2f + 1$ konsistente Stimmen zu sehen sind, wobei f die Anzahl der bösartigen Prüfer ist und die Gesamtzahl der Prüfer plus der Leiter $3f + 1$ ist. Die Commit-Phase beinhaltet einen ähnlichen Prozess der Stimmenauszählung, und ein Konsens wird erreicht, wenn $2f + 1$ konsistente Stimmen zu sehen sind. Aufgrund der erneuten Übertragung von Stimmen unter den Prüfern hat PBFT eine $O(N^2)$-Kommunikationskomplexität, die für ein Blockchain-System mit Hunderten oder Tausenden von Knoten (Nodes) nicht skalierbar ist.

Als Verbesserung von PBFT[14] ist das Konsensusprotokoll von Harmony in Bezug auf die Kommunikationskomplexität linear skalierbar, weshalb wir es Fast Byzantine Fault Tolerance (FBFT) nennen. In FBFT führt der Leiter (anstatt alle Prüfer aufzufordern, ihre Stimmen abzugeben), einen Multi-Signatur-Signierungsprozess durch, um die Stimmen der Prüfer in einer $O(1)$-großen Multi-Signatur zu sammeln und dann zu übertragen. Anstatt also $O(N)$-Signaturen zu empfangen, erhält jeder Prüfer nur eine Multisignatur, wodurch die Kommunikationskomplexität von $O(N^2)$ auf $O(N)$ reduziert wird.

Die Idee, Multisignatur in $O(1)$-Größe zu verwenden, ist inspiriert von ByzCoins BFT[15], das das Schnorr-Signaturschema für die Aggregation von Multisignaturen in konstanter Größe verwendet und einen Multicastbaum unter den Prüfern bildet, um die Nachrichtenzustellung zu erleichtern. Eine Schnorr-Multisignatur erfordert jedoch eine geheime Verpflichtungsrunde, was zu insgesamt zwei Rundläufen für eine einzige Multisignatur führt. Harmony verbessert dies durch die Verwendung von BLS (Boneh-Lynn-Shacham) Multisignatur [28], die nur einen Hin- und Rücklauf (einen Rundlauf) erfordert. Daher ist FBFT mindestens 50% schneller als das BFT von ByzCoin. Außerdem verwendet Harmony den RaptorQ Quellcode, um den Block Übertragungsprozess zu beschleunigen (siehe §6.2). Die Quellcode-Übertragungstechnik vermeidet auch ein Sicherheitsproblem im ursprünglichen baumbasierten Multicasting-Design von ByzCoin[16,17].

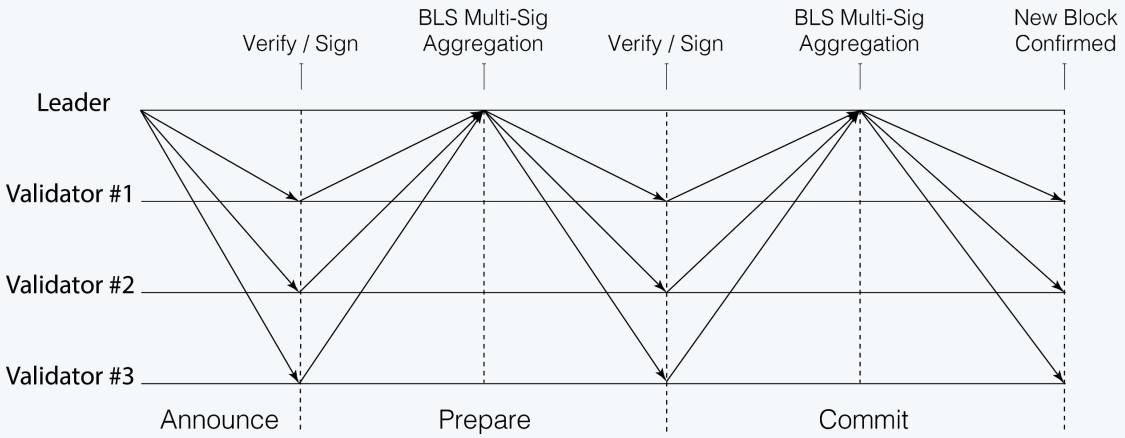

Abbildung 1. Netzwerkkommunikation während einer einzigen Konsensus Runde.

Insbesondere beinhaltet der FBFT-Konsens von Harmony die folgenden Schritte:

1. Der Leiter (Leader) erstellt den neuen Block und sendet den Blockkopf an alle Prüfer (Validator #1, #2, #3). In der Zwischenzeit sendet der Leiter den Inhalt des Blocks mit öschkodierung. (Details siehe §6.2). Dies wird als "announce"-Phase (Ankündigungsphase) bezeichnet.
2. Die Prüfer überprüfen die Gültigkeit des Blockkopfes, unterschreiben den Blockkopf mit einer BLS-Signatur und senden die Signatur an den Leiter zurück.
3. Der Leiter wartet auf mindestens $2f + 1$ gültige Signaturen von Prüfern (einschließlich des Leiters selbst) und fasst sie zu einer BLS-Multisignatur zusammen. Dann sendet der Leiter die aggregierte Multisignatur zusammen mit einer Bitmap, die angibt, welche Prüfer signiert haben. Zusammen mit Schritt 2 schließt dies die "Vorbereitungsphase" von PBFT ab.
4. Die Prüfer überprüfen, ob die Multisignatur mindestens $2f + 1$ Unterzeichner hat, überprüfen die Transaktionen in dem Blockinhalt, der vom Leiter in Schritt 1 übertragen wird, unterschreiben die empfangene Nachricht von Schritt 3 und senden sie an den Leiter zurück.
5. Der Leiter wartet auf mindestens $2f + 1$ gültige Signaturen (können verschiedene Unterzeichner aus Schritt 3 sein) aus Schritt 4, aggregiert sie zu einer BLS-Multisignatur und erstellt eine Bitmap, die alle Unterzeichner protokolliert. Schließlich überträgt der Leiter den neuen Block mit allen angehängten Multisignaturen und Bitmaps und sendet den neuen Block an alle Prüfer. Zusammen mit Schritt 4 schließt dies die "Commit"-Phase von PBFT ab.

Die Prüfer des Konsenses von Harmony werden auf der Grundlage des Proof-of-Stake gewählt. Daher unterscheidet sich das eigentliche Protokoll leicht von dem vorstehend beschriebenen in dem Sinne, dass ein Prüfer mit mehr Stimmrechtsanteilen mehr Stimmen hat als andere und nicht nur eine Unterschrift - eine Stimme. Anstatt also auf mindestens $2f + 1$ Unterschriften von Prüfern zu warten, wartet der Leiter auf Unterschriften von den Prüfern, die zusammen mindestens $2f + 1$

stimmberechtigte Anteile besitzen. Die Einzelheiten des Proof-of-Stake-Wahlmechanismus werden in §3.3 erläutert.

3. Sharding

Blockchain Sharding als Skalierbarkeitslösung hat seit Ende 2017 viel Aufmerksamkeit erregt. Sowohl in der Industrie als auch in der Wissenschaft wurden verschiedene Sharding-Lösungen vorgeschlagen.

In der Industrie war Zilliqa [12] die erste Sharding-basierte öffentliche Blockchain, die einen Durchsatz von 2.800 TPS erreichte. Zilliqa verwendet PoW als Identitätsregistrierungsverfahren (z.B. Sybil Angriff [1] Prävention). Das Netzwerk von Zilliqa besteht aus einem einzigen Verzeichnisdienstkomitee und mehreren Shard-Komitees *(z.B. Network Sharding)*, die jeweils Hunderte von Knoten (Nodes) enthalten. Transaktionen werden verschiedenen Shards zugeordnet und separat verarbeitet (z.B. Transaktions-Sharding). Die resultierenden Blöcke aus allen Shards werden im Verzeichnisdienstkomitee gesammelt und zusammengeführt. Zilliqa ist keine State-Sharding-Lösung, da jeder Knoten (jede Node) den gesamten Blockchain-Verlauf beinhalten muss, um Transaktionen verarbeiten zu können.

In der Wissenschaft haben Veröffentlichungen wie Omniledger [8] und RapidChain [7] Lösungen vorgeschlagen, die Zustands-Sharding beinhalten, bei dem jeder Shard eine Teilmenge des Blockchainzustands enthält. Omniledger verwendet ein Mehrparteien-Berechnungsschema namens RandHound [25], um eine sichere Zufallszahl zu erzeugen, die verwendet wird, um Knoten zufällig zu Shards zuzuordnen. Omniledger geht von einem *langsam anpassungsfähigen Korruptionsmodell* aus, bei dem Angreifer einen wachsenden Teil der Knoten in einem Shard im Laufe der Zeit beschädigen können. Unter einem solchen Sicherheitsmodell kann ein einzelner Shard schließlich beschädigt werden. Omniledger verhindert die Beschädigung von Shards, indem es alle Knoten (Nodes) in den Shards in einem festen Zeitintervall, der als “Epoche” bezeichnet wird, neu mischen kann. RapidChain baut auf Omniledger auf und schlägt die Verwendung der Bounded Cuckoo Rule (Regel des Gebundenen Kuckucks) vor, um Knoten (Nodes) ohne Unterbrechungen neu zu mischen [19].

Harmony lässt sich von diesen drei früheren Lösungen inspirieren [7,8,12] und entwirft ein PoS-basiertes Full-Sharding-System, das linear skalierbar und nachweislich sicher ist. Harmony enthält eine Signalkette (Beacon Chain) und mehrere Splitterketten (Shard-Chains). Die Signalkette dient als Zufallssignal und Identitäts Register, während die Shard-Ketten separate Blockchain-Zustände speichern und Transaktionen gleichzeitig verarbeiten. Harmony bietet einen effizienten Algorithmus zur Zufallsgenerierung durch die Kombination von Verifiable Random Function (VRF) und Verifiable Delay Function (VDF). Harmony integriert auch PoS in den Sharding-Prozess, was die Sicherheitsüberlegung eines Shards von der minimalen Anzahl der Knoten [7,8,12] auf die minimale Anzahl der Stimmrechte verschiebt.

3.1 Verteilte Zufallsgenerierung

Hintergrund

Verschiedene Ansätze wurden vorgeschlagen, um Knoten (Nodes) zu Shards zuzuordnen, wie z.B. zufallbasiertes Sharding [7,8], ortsbezogenes Sharding [34] und zentral gesteuertes Sharding [35]. Von allen Ansätzen wurde das zufällige Sharding als die sicherste Lösung auserwählt. Beim zufälligen Sharding wird eine gemeinsam vereinbarte Zufallszahl verwendet, um die Sharding-Zuordnung für jeden Knoten (jede Node) zu bestimmen. Die Zufallszahl muss folgende Eigenschaften aufweisen:

1. Unvorhersehbar: Niemand sollte in der Lage sein, die Zufallszahl vorherzusagen, bevor sie erzeugt wird.
2. Unbefangen: Der Prozess der Generierung der Zufallszahl sollte für keinen Teilnehmer eine beeinträchtigung sein.
3. Überprüfbar: Die Gültigkeit der erzeugten Zufallszahl sollte von jedem Beobachter nachprüfbar sein.
4. Skalierbar: Der Algorithmus der Zufallsgenerierung sollte auf eine große Anzahl von Teilnehmern skaliert werden.

Omniledger [8] verwendet das RandHound [25]-Protokoll, das ein führungs orientierter Prozess der verteilten Zufallsgenerierung (DRG) ist, der PVSS (Publicly Verifiable Secret Sharing) und byzantinische Vereinbarungen beinhaltet. RandHound ist ein $O(n * c^2)$ -Protokoll, das Teilnehmerknoten (teilnehmende Nodes) in mehrere Gruppen der Größe c unterteilt. Es erfüllt die ersten drei der oben genannten Eigenschaften, ist aber unpraktisch und langsam, um sich als skalierbar zu qualifizieren.

RapidChain [7] verfolgt einen einfacheren Ansatz, indem es jeden Teilnehmer VSS (Verifiable Secret Sharing) [22] durchführen lässt und die kombinierten geheimen Anteile als Zufallsgenerator verwendet. Leider ist dieses Protokoll nicht sicher, da die bösartigen Knoten (Nodes) inkonsistente Freigaben an verschiedene Knoten (Nodes) senden können [25]. Außerdem beschreibt RapidChain nicht, wie die Knoten (Nodes) einen Konsens über die verschiedenen möglichen Versionen der rekonstruierten Zufälligkeit erreichen.

Darüber hinaus stützt sich Algorand [18] auf die VRF-basierte (Verifiable Random Function) kryptographische Sortierung; um die Gruppe der Konsensus-Prüfer auszuwählen. Das Design von Ethereum 2.0 schlägt den Einsatz von VDF (Verifiable Delay Function) [20] vor, um die Offenbarung der tatsächlichen Zufallszahl zu verzögern und so einen Angriff des “letzten Enthüllers” zu verhindern [36]. Das VDF ist ein neu erfundenes kryptographisches Primitiv; es benötigt eine einstellbare Mindestzeit für die Berechnung und das Ergebnis kann sofort verifiziert werden.

Skalierbare Zufallsgenerierung mit VRF und VDF

Der Ansatz von Harmony kombiniert die Stärken der oben genannten Lösungen. Erstens ist die Komplexität des DRG-Protokolls von Harmony $O(n)$, das in der Praxis mindestens eine Größenordnung schneller ist als RandHound. Zweitens ist unser Ansatz im Gegensatz zu RapidChains einfachem VSS-basiertem Ansatz unvoreingenommen und verifizierbar. Drittens, im Vergleich zur Lösung von Ethereum 2.0, verwendet unser Ansatz den BFT-Konsens, um der Zufallszahl Endgültigkeit zu verleihen. Im Einzelnen beinhaltet das Protokoll die folgenden Schritte:

1. Ein Leiter sendet eine Init-Message mit dem Hash des letzten Blocks $H(B_{n-1})$ an alle Prüfer.
2. Für jeden Prüfer i wird nach dem Empfangen der Init-Nachricht ein VRF berechnet, um eine Zufallszahl r_i und einen Beweis p_i zu erzeugen: p_i: $(r_i, p_i) = VRF(sk_i, H(B_{n-1}), v)$, wobei sk_i der geheime Schlüssel des Prüfers i ist und v die aktuelle View-Nummer des Konsenses ist. Dann sendet jeder Prüfer (r_i, p_i) an den Leiter zurück.
3. Der Leiter wartet, bis er mindestens $f + 1$ gültige Zufallszahlen erhält und kombiniert sie mit einer XOR-Operation, um das Vorbild der endgültigen Zufälligkeit pRnd zu erhalten.
4. Der Leiter leitet BFT (diskutiert in §2) zu allen Prüfern, um einen Konsens über das $pRnd$ zu erzielen und es in Block B_n zu binden.
5. Nachdem pRnd committed wurde, beginnt der Leiter mit der Berechnung der tatsächlichen Zufälligkeit $Rnd = VDF(pRnd, T)$, wobei T die VDF-Schwierigkeit ist und algorithmisch so eingestellt ist, dass die Zufälligkeit erst nach k Blöcken berechnet werden kann.
6. Sobald Rnd berechnet ist, initiiert der Leiter ein BFT unter allen Prüfern, um sich auf die Gültigkeit von Rnd zu einigen und schließlich die Zufälligkeit in die Blockkette zu übertragen.

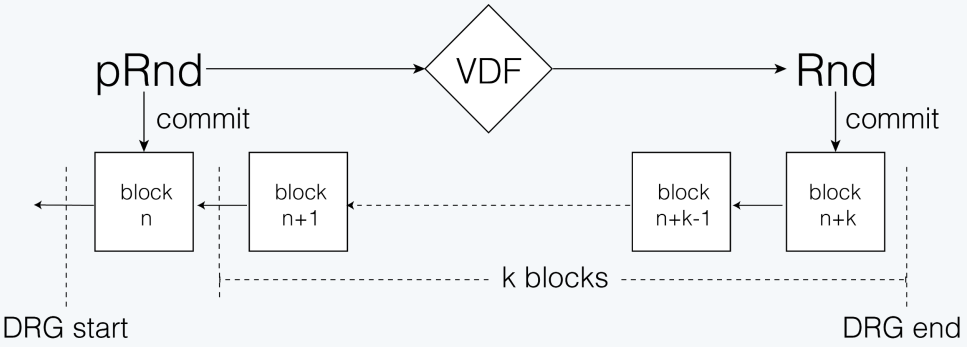

Abbildung 2. Die VDF (Verifiable Delay Function) verzögert die Veröffentlichung der endgültigen Zufälligkeit.

Die VDF wird verwendet, um die Veröffentlichung von Rnd nachweislich zu verzögern und zu verhindern, dass ein böswilliger Leiter (Leader) die Zufälligkeit beeinträchtigt, indem er eine Teilmenge der VRF-Zufallszahlen auswählt. Aufgrund der VDF wird der Leiter (Leader) nicht in der Lage sein, die tatsächliche endgültige Zufälligkeit zu kennen, bevor $pRnd$ an die Blockchain gebunden wird. Wenn Rnd mit dem VDF berechnet wird, ist $pRnd$ bereits in einem vorherigen Block gebunden, so dass der Leiter ihn nicht mehr manipulieren kann. Daher ist das Beste, was ein

bösartiger Leiter (Leader) tun kann, entweder blind die Zufälligkeit $pRnd$ zu übertragen oder das Protokoll zu blockieren, indem er $pRnd$ nicht verpflichtet. Ersteres ist das gleiche wie das ehrliche Verhalten. Letzteres verursacht keinen großen Schaden, da der gleiche Timeout-Mechanismus in PBFT [14] verwendet wird, um den Leiter zu wechseln und das Protokoll neu zu starten.

Wir gehen langfristig von der Existenz von ASICs zur Berechnung von VDFs aus, bei denen einige altruistische Knoten (Nodes) mit einem ASIC (Application-Specific Integrated Circuit) das Ergebnis veröffentlichen und niemand das System spielen konnte. Es ist möglich, dass ein Angreifer mit einem schnelleren Computer Gerät das Ergebnis vor anderen ehrlichen Knoten (Nodes) berechnen kann, bevor VDF-ASICs in Produktion sind. Bis dies geschieht, kann der Angreifer den Zufall nur kurz vor den ehrlichen Knoten (Nodes) erkennen. Während der Angreifer dies grundsätzlich nutzen könnte (z.B. die Auszahlung seines Kapitals, wenn die Wette auf einen Smart Contract für ihn ungünstig war), kann dieses Problem auf der Smart Contract Ebene mit einer angemessenen Verzögerung gemildert werden, so dass es eine Wartezeit geben sollte, bis die Zufälligkeit auf das Protokoll übertragen wird, bevor eine Auszahlung des Kapitals ermöglicht wird.

3.2 Epochen

In Harmony wird der Konsensus- und Sharding-Prozess durch das Konzept der Epochen orchestriert. Eine Epoche ist ein vorgegebenes Zeitintervall (z.B. 24 Stunden), in dem die Sharding-Struktur fixiert ist und jeder Shard kontinuierlich mit dem gleichen Satz von Prüfern übereinstimmt. Zu Beginn jeder Epoche wird mit dem in §3.1 beschriebenen DRG-Protokoll eine Zufallszahl erzeugt und die Sharding-Struktur basierend auf dieser Zufälligkeit bestimmt. Prüfer, die Transaktionen in der Epoche bestätigen wollen, müssen ihre Token während der Epoche $e - 1$ setzen. Die Ausschaltzeit für das Staken (stapeln) liegt vor dem Einbinden des Zufalls-Vorbildes $pRnd$ in die Blockchain.

3.3 Staking-basiertes Sharding

Registrierung des Prüfers

Die Prävention von Sybil-Angriffen [1] ist ein wichtiger Sicherheitsaspekt in öffentlichen Blockchains. Bitcoin und Ethereum verlangen von den Minern, dass sie ein kryptographisches Puzzle (PoW) berechnen, bevor sie einen Block vorschlagen können. Ebenso verwenden Sharding-basierte Blockchains wie Zilliqa [12] oder Quarkchain [11] ebenfalls PoW, um Sybil-Angriffe zu verhindern. Harmony verfolgt einen anderen Ansatz mit Proof-of-Stake (PoS) als Validator-Registrierung oder Sybil-Angriffsverhinderungsmechanismus. Um ein Harmony Prüfer zu werden, müssen potenzielle Teilnehmer (oder Staker /Stapler) eine bestimmte Anzahl von Token setzen, um berechtigt zu sein. Die Anzahl der eingesetzten Token bestimmt die Anzahl der dem Prüfer zugewiesenen Stimmrechtsanteilen. Jeder Stimmrechtsanteil entspricht einer Stimme im BFT-Konsens (wie in §2 erläutert).

Beteiligung durch Stimmrechte

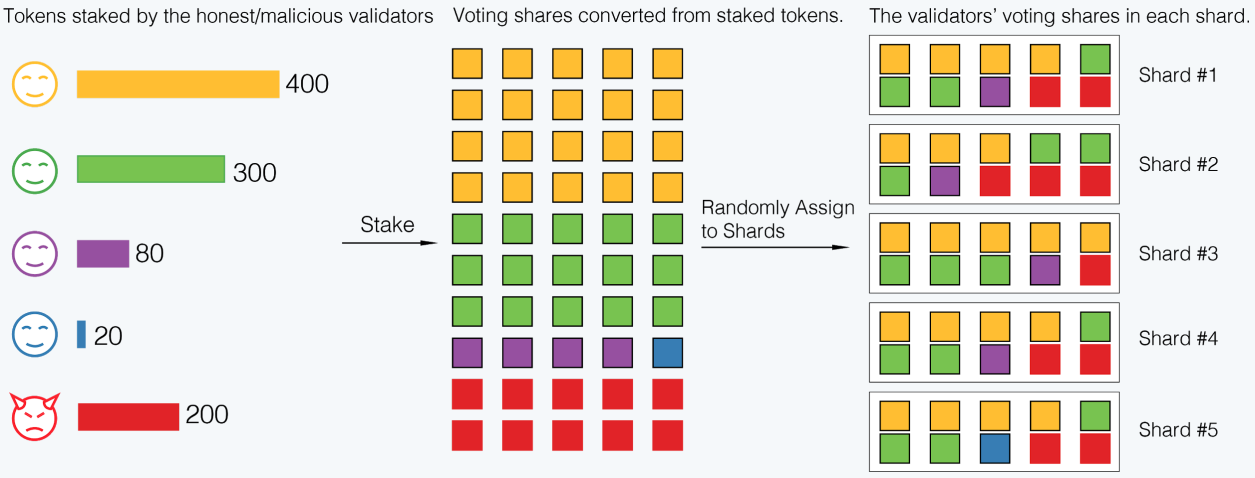

Abbildung 3. Die Staker /Stapler erhalten Stimmrechte proportional zu ihren gestaffelten Token. Die Stimmrechte werden dann zufällig auf Shards verteilt. Staker /Stapler werden zu Prüfern für die Shards, bei denen ihre Stimmrechte zugewiesen werden.

Ein Stimmrechtsanteil ist ein virtuelles Ticket, das es einem Prüfer ermöglicht, eine Stimme im Konsens abzugeben. Prüfer können Stimmrechtsanteile durch Staken /Stapeln von Token erwerben. Die Anzahl der für einen Stimmrechtsanteil benötigten Token wird algorithmisch angepasst. Zu Beginn jeder Epoche werden die Stimmrechte neuer Prüfer nach dem Zufallsprinzip auf Shards verteilt. Die neuen Prüfer schließen sich den Shards an, in denen ihre Stimmrechte zugewiesen werden. Wie in §2 besprochen, wird der Konsens in einem Shard von Prüfern erreicht, die zusammen mindestens $2f + 1$ stimmberechtigte Anteile besitzen, um den Block zu unterzeichnen..

Um die Sicherheit eines einzelnen Shards zu gewährleisten, muss die Anzahl der Stimmrechte durch böswillige Prüfer unter $\frac{1}{3}$ aller Stimmrechte an diesem Shard gehalten werden. Dies ist aufgrund der Art des BFT-Konsenses erforderlich. Der adaptive Thresholded PoS von Harmony garantiert die oben genannte Sicherheitsanforderung, indem er den Preis einer Stimmrechtsaktie adaptiv anpasst und einzelne Stimmrechtsaktien eher Shards als einzelnen Prüfern zuordnet.

Unsere Sicherheits Annahme ist, dass über alle eingesetzten Token hinweg bis zu $\frac{1}{4}$ davon bösartigen Prüfern gehören. Wenn wir durch Prüfer sharden (d.h. einen Prüfer einem Shard zuordnen), im schlimmsten Fall, wenn ein einzelner bösartiger Prüfer $\frac{1}{4}$ aller gesteigerten Token (oder der stimmberechtigten Anteile) hält, wird er leicht mehr als $\frac{1}{3}$ stimmberechtigte Anteile in diesem Shard besitzen. Der Grund dafür ist, dass die Einsätze an jedem Shard "m" mal kleiner sind als die Einsätze des gesamten Netzwerks, wobei "m" die Anzahl der Shards ist. Wir nennen dieses Angriffsszenario einen Großangriff (eine spezielle Form des Single-Shard Takeover-Angriffs).

Um Angriffe mit hohen Einsätzen zu verhindern, werden anstelle von Shards durch Prüfer, Shards durch Stimmanteile vergeben (d. H. Ein Shard wird ein Stimmanteil zugewiesen). Insbesondere wird, nachdem das Rnd zu Beginn der aktuellen Epoche enthüllt wurde, eine zufällige (mit Rnd gesetzte) Permutation für alle stimmberechtigten Anteile durchgeführt, und die permutierte Liste der stimmberechtigten Anteile wird gleichmäßig in "m" Töpfen aufgeteilt, wobei "m" die Anzahl der Shards ist. Die stimmberechtigten Anteile, die in den "i" Töpfen fallen, werden dem Shard "i" zugewiesen, ebenso die entsprechenden Prüfer. In der Praxis kann ein einzelner Prüfer mehreren Shards zugewiesen werden, wenn er über diesen Shards zugewiesene Stimmrechtsanteile verfügt. Der Shard-Leiter wird als der Prüfer bestimmt, der den ersten Stimmanteil im Topf besitzt.

Es ist zu erwähnen, dass Prüfer mit größeren Einsätzen mehr Chancen haben, als Leiter ausgewählt zu werden. Wir sind der Ansicht, dass dies tatsächlich ein wünschenswertes Szenario ist, da große Akteure aufgrund der Befürchtung einer Kürzung ihres Einsatzes einen größeren Anreiz haben, dem Protokoll zu folgen (der Anreizmechanismus wird in §7 erörtert). Darüber hinaus verfügen sie mit größerer Wahrscheinlichkeit über leistungsstärkere Maschinen mit einem schnellen und stabilen Netzwerk.

Adaptiver- Schwellenwert PoS

Der Preis eines stimmberechtigten Anteils wird algorithmisch so festgelegt, dass er klein genug ist, dass böswillige Staker /Stapler ihre Stimmrechte nicht auf einen einzigen Shard konzentrieren können. Konkret setzen wir den Preis eines stimmberechtigten Anteils auf P_{vote}-Token fest:

$$P_{vote} = \frac{TS_{e-1}}{NumShard^{*}\,\lambda}$$

Hier ist λ ein Sicherheitsparameter, $NumShard$ ist die Anzahl der Shards und TS_{e-1} ist die Gesamtzahl der Token, die während der Epoche $e-1$ eingesetzt wurden

Jetzt beweisen wir, dass, wenn $\lambda > 600$, die Wahrscheinlichkeit, dass ein einzelner Shard mehr als $\frac{1}{3}$ böswillige Stimmrechte (d.h. die Wahrscheinlichkeit des Scheiterns) hat, vernachlässigbar ist.

Nach der Definition von P_{vote}, wird die Gesamtzahl der Stimmrechte $N = \frac{TS_{e-1}}{P_{vote}} = NumShard * \lambda$. Betragen.

Bei einer vertrauenswürdigen Zufallsquelle (diskutiert in §3.1) und dem auf der Zufälligkeit basierenden Sharding-Prozess kann die Wahrscheinlichkeitsverteilung der Anzahl der bösartigen Stimmrechtsanteile in jedem Shard als hypergeometrische Verteilung modelliert werden (d.h. Stichproben ohne Ersatz):

$$P(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

Hier ist N die Gesamtzahl der stimmberechtigten Anteile, $K = \frac{N}{4}$ ist die maximale Anzahl der bösartigen Stimmrechte, $n = \frac{N}{NumShard}$ ist die Anzahl der Stimmrechte in jedem Shard und k ist die Anzahl der bösartigen Stimmrechte in einem Shard. Die tatsächliche Ausfallrate eines Shards $P\ (X \leq k)$ folgt der kumulativen hypergeometrischen Verteilung $CDF_{hg}(N,\ K,\ n,\ k)$, die, wenn N groß ist, zu einer binomialen Verteilung (d.h. Stichproben mit Ersatz) abnimmt:

$$P(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

Wir zeigen hier, dass, wenn "n" groß genug ist, die Wahrscheinlichkeit, dass ein Shard mehr als $\frac{1}{3}$ Token enthält, die von bösartigen Entitäten gehalten werden, vernachlässigbar ist. Tatsächlich ist die Wahrscheinlichkeit, dass ein Shard weniger als $\frac{1}{3}$ der bösartigen Stimmrechte enthält, bei $P(X \leq 200) = 0.999997$, was zu einer Shard-Fehlerrate von "einmal in etwa 1000 Jahren" (bei einem Epochen Intervall von 24 Stunden) führt. Deshalb werden wir $\lambda = 600$ einstellen, um die hohe Sicherheit unserer Shards zu gewährleisten. (Intuitiv regelt λ die Mindestanzahl der Stimmrechte, die ein einzelner Shard enthalten sollte. Dies ist funktional ähnlich wie die minimale Anzahl von Knoten (Nodes) in einem Shard, wie in anderen PoW-basierten Sharding-Lösungen beschrieben [7,8,12]).

Dieser Ansatz ist resistent gegen die Schwankungen der Anzahl der Prüfer. Wir setzen keine untere Grenze für die Anzahl der Prüfer in jedem Shard wie in anderen Lösungen wie Zilliqa [12]. Stattdessen wenden wir ein adaptives PoS-basiertes Modell an, um sicherzustellen, dass die bösartigen Menschen niemals mehr als $\frac{1}{3}$ der Stimmrechte in einem einzigen Shard einnehmen können, und machen es so sicher.

3.4 Resharding

Wir haben ein sicheres Sharding-Schema beschrieben, das verhindert, dass böswillige Prüfer einen einzelnen Shard übernehmen. Wenn die Sharding-Struktur jedoch fixiert bleibt, können böswillige Angreifer immer noch einen Shard übernehmen, indem sie die Prüfer in diesem Shard beschädigen. Es gibt drei Modelle von Angreifern:

1. Statische Rund-Adaption: Hier können Angreifer nur eine Teilmenge von Knoten (Nodes) in einem vorgegebenen Stadium beschädigen. Elastico [9] geht davon aus, dass Angreifer nur zu Beginn jeder Epoche Knoten (Nodes) beschädigen können.

2. Langsam anpassungsfähig: Hier können Angreifer im Laufe der Zeit während der Epoche eine Teilmenge von Knoten (Nodes) beschädigen [7,8].
3. Vollständig anpassungsfähig: wo Angreifer eine Teilmenge von Knoten (Nodes) sofort und jederzeit beschädigen können [18]

Harmony geht von dem langsam anpassungsfähigen Korruptionsmodell aus, bei dem der Angreifer eine konstante Anzahl von Knoten (Nodes) beschädigen kann und es eine gewisse Zeit dauert. Omniledger [8] geht von demselben Korruptionsmodell aus und verhindert den Angriff, indem es in jeder Epoche Prüfer in allen Shards ersetzt. Dieser Ansatz hat zwei Hauptprobleme. Das erste sind die hohen Kosten für das “bootstrapping” in jeder Epoche. Die zweite ist das Sicherheitsproblem, wenn alle Knoten (Nodes) während des Konsenses ersetzt werden.

Harmony mildert diese Probleme, indem es den auf der Cuckoo-rule (Kuckucksregeln) basierenden Resharding-Mechanismus [7,19] anwendet. Nach dem Ende einer Epoche werden die Prüfer, die ihren Einsatz zurückgezogen haben, aus dem Netzwerk vertrieben, während diejenigen, die ihren Einsatz behalten, bleiben. Die neuen Prüfer, die in dieser Epoche etwas eingesetzt haben, erhalten neue Stimmrechte. Diese Stimmrechte werden nach dem Zufallsprinzip den Anteilen zugeordnet, die mehr als den Median der gesamten Stimmrechte haben. Als nächstes wird eine konstante Anzahl der stimmberechtigten Aktien aus allen Shards zufällig auf die andere Hälfte der Shards umverteilt, die weniger als den Median der gesamten stimmberechtigten Aktien haben. Es ist in [7] bewiesen, dass dieses Auffrischungsschema die Stimmanteile in allen Shards im Gleichgewicht halten und gleichzeitig die Sicherheitsanforderungen erfüllen kann.

3.5 Schnelle Zustands-Synchronisation

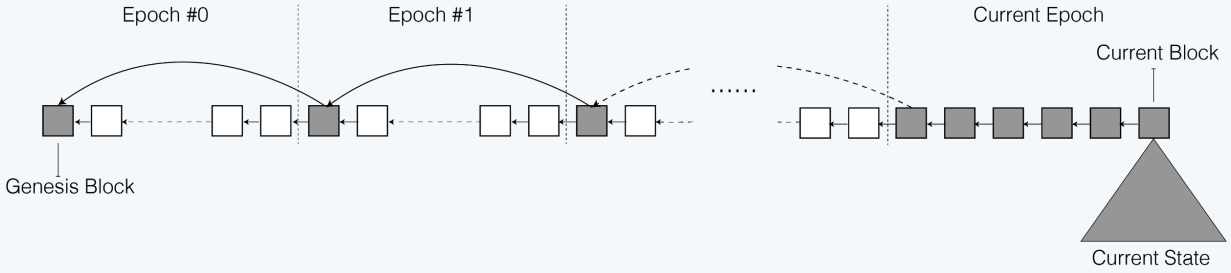

Abbildung 4. Der erste Block einer Epoche enthält einen Hash-Link zum ersten Block der letzten Epoche. Dies ermöglicht eine schnelle Zustands-Synchronisation neuer Knoten, bei der sie sich nur auf die grau markierten Blöcke verlassen können, um den aktuellen Zustand schnell zu überprüfen.

Wenn Prüfer einem neuen Shard beitreten, müssen sie sich schnell auf den aktuellen Zustand des Shards synchronisieren, um neue Transaktionen zu bestätigen. Das traditionelle Verfahren zum Herunterladen der Blockchain-Historie und zum Rekonstruieren des aktuellen Zustands ist zu langsam, um eine Neufassung zu ermöglichen (es dauert Tage, bis die Ethereum-Blockchain-Historie vollständig synchronisiert ist). Glücklicherweise ist der aktuelle Zustand um Größenordnungen kleiner als die gesamte Blockchain-Geschichte. Das Herunterladen

des aktuellen Zustands innerhalb des Zeitfensters einer Epoche ist möglich, im Vergleich zum Herunterladen der gesamten Historie.

In Harmony laden neue Prüfer, die einem Shard beitreten, zunächst den aktuellen Status dieses Shards herunter, damit sie mit der Bestätigung von Transaktionen schnell beginnen können. Um sicherzustellen, dass der aktuelle Status des Downloads gültig ist, muss der neue Knoten (die neue Node) eine ordnungsgemäße Überprüfung durchführen. Anstatt die gesamte Blockchain-Historie herunterzuladen und alle Transaktionen erneut abzuspielen, um den aktuellen Zustand zu überprüfen, lädt der neue Knoten (die neue Node) historische Blockheader herunter und überprüft die Header, indem er ihre Signaturen überprüft. Solange es eine kryptografische Spur (z.B. Hashpointer und Signaturen) vom aktuellen Zustand bis zum Genesis-Block (erster Block der Blockchain) gibt, ist der Zustand gültig. Dennoch ist die Signaturprüfung nicht rechnerisch kostenlos und es dauert sehr lange, bis alle Signaturen ab dem Genesis-Block überprüft sind. Um dieses Problem zu mildern, enthält der erste Block jeder Epoche einen zusätzlichen Hash-Pointer auf den ersten Block der letzten Epoche. Auf diese Weise kann der neue Knoten innerhalb einer Epoche über die Blöcke springen, wenn er Hashpointer auf den Genesis-Block verfolgt. Dies wird die Überprüfung des aktuellen Blockchain-Zustandes erheblich beschleunigen.

Um den Zustands-Synchronisationsprozess weiter zu optimieren, werden wir den Blockchain-Zustand selbst so Klein wie möglich gestalten. Eine Beobachtung aus dem Ethereum-Blockchain Zustand ist, dass viele Konten (Nodes) leer sind und den kostbaren Raum des Blockchain-Zustands verschwenden. In Ethereum können die leeren Konten (Nodes) mit einem bestimmten Nonce nicht gelöscht werden, da es sich um potenzielle Replay-Angriffe handelt, bei denen alte Transaktionen auf dem gelöschten Konto erneut übertragen werden [32]. Harmony verwendet ein anderes Modell zur Vermeidung von Replay-Angriffen, indem die Transaktionen den Hash des aktuellen Blocks angeben: Eine Transaktion ist nur gültig, wenn eine bestimmte Anzahl (z.B. 100) von Blöcken auf den Block des angegebenen Hash folgt. Auf diese Weise können die alten Konten sicher gelöscht und der Blockchain-Status schlank gehalten werden.

4. Shard Chain und Beacon Chain

4.1 Shard Chain

Eine Shard Chain ist eine Blockchain, die ihre eigenen Transaktionen verarbeitet und überprüft und ihren eigenen Zustand speichert. Ein Shard verarbeitet nur Transaktionen, die für sich selbst relevant sind. Obwohl eine Shard-Chain relativ unabhängig ist, kommuniziert sie mit anderen Shard-Chains durch Cross-Shard-Kommunikation.

Shardübergreifende Kommunikation

Cross-Shard-Kommunikation ist eine Schlüsselkomponente jeder Sharding-basierten Blockchain. Die Cross-Shardfähigkeit durchbricht die Barriere zwischen den Shards und erweitert den Nutzen eines einzelnen Shards. Insgesamt gibt es drei Kategorien der Cross-Shard- Kommunikation:

1. Main-chain-driven: Projekte wie Zilliqa [12] verlassen sich auf die Hauptkette, um Transaktionen über Shards hinweg zu erreichen.
2. Client-driven: Omniledger [8] schlug einen mandanten gesteuerten Cross-Shard-Transaktionsmechanismus vor, bei dem die Nachrichten zwischen den Shards gesammelt und von Clients an Shards gesendet werden. Dies bedeutet eine zusätzliche Belastung für den Client, die für einen Adhoc-Light-Client nicht wünschenswert ist.
3. Shard-driven: RapidChain [7] schlug vor, dass die Nachrichten zwischen den Shards direkt von den Knoten im Shard ohne externe Hilfe gesendet werden.

Harmony verfolgt den o.g. shard-driven Ansatz wegen seiner Einfachheit und der Abwesenheit von Belastungen für die Kunden. Wir sind der Meinung, dass die Vorteile der teilnehmer orientierten Kommunikation die Nachteile überwiegen. Die Kosten für die shard-driven Kommunikation im Gesamtnetzwerk können beträchtlich sein, da jede Cross-Shard-Nachricht eine Broadcast auf Netzwerkebene ist, die $O(N)$ Netzwerkkosten verursacht. Um dieses Problem zu lösen, verwendet Harmony das Kademlia-Routing-Protokoll, um die Komplexität der Kommunikation auf $O(log(N))$ zu reduzieren. Darüber hinaus werden die übermittelten Daten mit einem Löschcode kodiert; um die Robustheit der Cross-Shard- Kommunikation zu gewährleisten. Die Details werden in §6 erläutert.

4.2 Beacon Chain

Die Harmony Beacon-Chain ist eine spezielle Blockchain, die im Vergleich zu den Shard-Chains zusätzliche Funktionen erfüllt. In der Tat ist die Beacon-Chain auch ein Shard. Neben der Verarbeitung von Transaktionen, wie es andere Shard-Chains tun, übernimmt die Beacon-Chain zwei weitere Schlüsselfunktionalitäten: die Generierung der Zufallszahl (siehe §3.1) und die Annahme von Stakes, was bedeutet, dass die Beacon-Chain die Kette ist, in der die Stakeholder ihre Token hinterlegen, um Prüfer zu werden.

Die Prüfer für die Beacon-Chain werden ähnlich wie die anderen Shard-Chains bestimmt. Während der Shardingzuweisung werden die Stimmanteile zufällig in $NumShard + b$ Töpfe aufgeteilt, wobei die zusätzlichen b Töpfe für die Beacon-Chain sind.

Hash Link von der Shard Chain

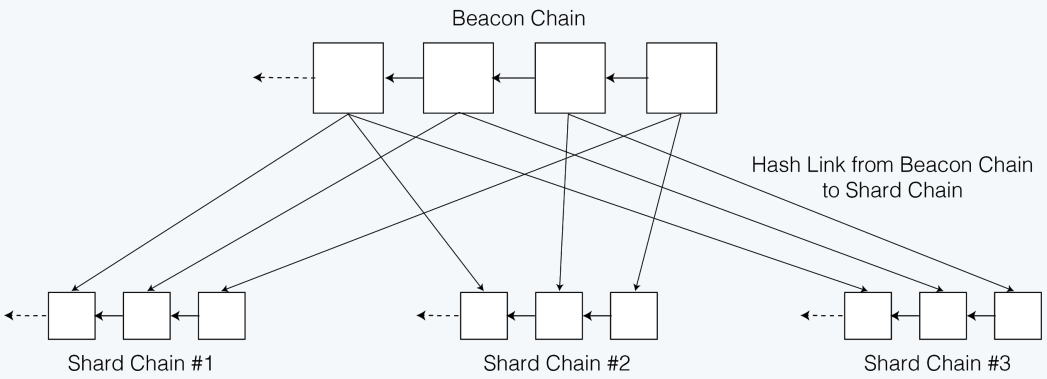

Abbildung 5. Hash-Link vom Beacon-Chainblock zum Shard-Chainblock.

Die Beacon-Chain trägt dazu bei, die Sicherheit und Konsistenz der Zustände der Shard-Chains zu erhöhen, indem sie den Blockkopf von jeder Shard-Chain einbezieht. Insbesondere, nachdem ein neuer Block an eine Shard-Chain gebunden wurde, wird sein Block-Header (über eine Kademlia-basierte Inter-Shard-Kommunikation) an die Beacon-Chain gesendet. Die Beacon-Chain überprüft die Gültigkeit des Blockkopfes durch:

1. Der Hash seines vorherigen Blocks, der bereits in der Beacon-Chain abgelegt worden sein muss;
2. Die Unterzeichner der Multisignatur des Blocks, die die richtigen Prüfer für diesen Shard sein müssen.

Die zugesagten Block-Header an der Beacon-Chain werden dann an das gesamte Netzwerk übertragen. Jeder Shard behält eine Kette von gültigen Block-Headern für alle anderen Shards, mit denen die Gültigkeit von Transaktionen aus anderen Shards überprüft wird (z.B. einfache Zahlungsprüfung). Das Hinzufügen der Blocksammler der Shard-Chains in die Beacon-Chain dient vor allem zwei Zwecken:

1. Erhöht die Schwierigkeit, einen einzelnen Shard anzugreifen. Angreifer müssen sowohl die Shard-Chain als auch die Beacon-Chain beschädigen, um andere davon zu überzeugen, dass ein alternativer Block in der Shard-Chain gültig ist.
2. Reduzierung der Netzwerkkosten für die Übertragung der Block-Header zwischen den Shards. Es wird eine $O(N^2)$ -Netzwerkkommunikation geben, wenn wir jeden Shard seine Header separat senden lassen. Mit der Beacon-Chain als zentrales Relais reduziert sich die Komplexität auf $O(N)$.

5. Blockchain State-Sharding

Im Gegensatz zu anderen State-Sharding-Blockchains [7,8], die das UTXO-Datenmodell (Unspent Transaction Output) übernommen haben, wird Harmony's State Sharding auf das kontenbasierte

Datenmodell angewendet. Jede Shard-Chain enthält ihren eigenen Kontostand, und alle vorhandenen Token sind auf alle Shards verteilt.

Wir behandeln das Benutzerkonto und das Smart Contract Account beim Sharding unterschiedlich. Ein Benutzerkonto kann mehrere Salden auf verschiedenen Shards haben (z.B. 100 Token auf Shard A und 50 Token auf Shard B). Ein Benutzerkonto kann seinen Saldo zwischen den Shards verschieben, indem es eine Cross-Shard- Transaktion durchführt. Ein Smart Contract Account ist auf den spezifischen Shard beschränkt, in dem der Vertrag angelegt wurde. Bei einer dezentralen Anwendung, die mehr Durchsatz erfordert, als ein einzelner Shard verarbeiten kann, kann der Dapp-Entwickler (Dezentrale Anwendung) jedoch mehrere Instanzen desselben Smart Contract in verschiedenen Shards instanziieren und jede Instanz eine Teilmenge des eingehenden Datenverkehrs verarbeiten lassen. Zu beachten ist, dass sich die verschiedenen Instanzen desselben Smart Contract nicht den gleichen Zustand teilen, aber sie können über Cross-Shard-Kommunikation miteinander kommunizieren.

6. Vernetzung

Frühere Untersuchungen [33] haben gezeigt, dass die Netzwerkkapazität einer der größten Engpässe für Blockchain-Systeme ist. Um die Leistung zu steigern, konzentriert sich Harmony auf die Verbesserung der Effizienz der Netzwerknutzung. Harmony schlägt auch eine Reihe von Verbesserungen vor, um mit realen Netzwerk Szenarien umzugehen.

6.1 Kademlia-basiertes Routing

Inspiriert von RapidChain [7] werden wir Kademlia [37] als Routing-Mechanismus für Cross-Shard-Meldungen übernehmen. Jeder Knoten im Harmony-Netzwerk unterhält eine Routingtabelle, die Knoten aus verschiedenen Shards enthält. Der Abstand zwischen den Shards ist definiert als der XOR-Abstand der Shard IDs. Wenn eine Nachricht von Shard A an Shard B gesendet werden muss, schauen sich die Knoten (Nodes) in Shard A die Routingtabelle an und senden die Nachricht an die Knoten (Nodes) mit der nächstgelegenen Shard ID. Beim Kademlia-basierten Routing wandert eine Nachricht nur über $O(logN)$-Knoten, bevor sie den Ziel-Shard erreicht. Im Vergleich zu normalen Rundsendungen, die eine $O(N)$ -Netzwerk-Komplexität erfordern, kann der Kademlia-Routing-Mechanismus die Gesamtnetzlast in einer sharded Blockchain deutlich reduzieren.

6.2 Effiziente Übertragung mit Löschcode

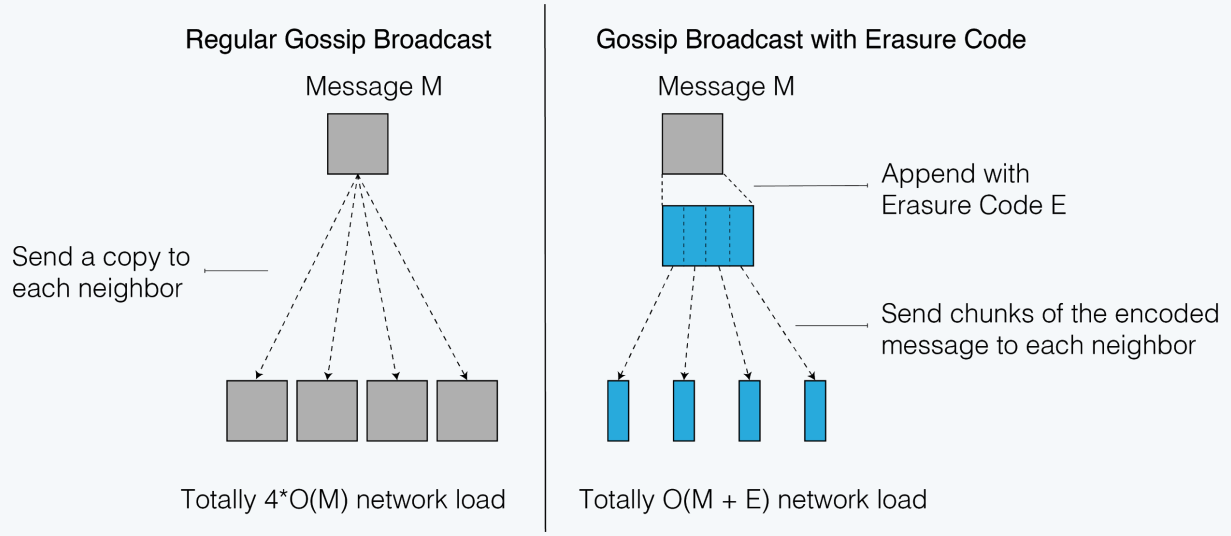

Abbildung 6. Vergleich zwischen normaler “Gerüchteverbreitung” und “Gerüchteverbreitung” mit Löschcode.

Broadcast ist eine häufige Netzwerkaktion in jedem Blockchain-System, das auf P2P (Peer-to-Peer) Netzwerk-Overlay basiert. Konkret in unserem Konsens Protokoll gibt es drei Szenarien, in denen eine Übertragung erforderlich ist:

1. Ein neu vorgeschlagener Block muss vom Leiter an alle Prüfer gesendet werden.
2. Ein neu generierter Master-Chainblock muss an das gesamte Netzwerk gesendet werden.
3. Die Cross-Shard-Kommunikation erfordert die Übertragung einer Nachricht zwischen den Shards.

Bei einer normalen P2P-Übertragung muss der ursprüngliche Absender eine Kopie der Nachricht an jeden seiner Nachbarn senden. Dies führt zu einer $O(d * M)$ -Netzwerklast auf dem Sender, wobei d die durchschnittliche Anzahl der Nachbarn des Senders und M die Nachrichtengröße ist. Stattdessen kodiert ein Absender in Harmony zuerst die Nachricht mit einem Löschcode und sendet dann Teile der kodierten Nachricht an jeden Nachbarn. Dies reduziert die Belastung des Absenders auf $O(M + e)$, wobei e die Größe des Löschcodes ist und normalerweise kleiner als die Größe der ursprünglichen Nachricht M ist. Daher senkt der Netzwerkbetrieb-Mechanismus von Harmony die Netzwerklast des Absenders erheblich. Darüber hinaus schlägt Harmony vor, die Robustheit von IDA zu verbessern, indem der ursprüngliche Reed-Solomon Löschcode durch RaptorQ Quellcode ersetzt wird, so dass der Sender immer mehr Löschcodes senden kann, um sicherzustellen, dass die Daten schließlich empfangen werden.

6.3 FEC-basierter Unicast

Traditionelle zuverlässige Transporte wie TCP [42] basieren auf Retransmission und ACK-basierter Signalisierung, um mit verlorenen Paketen umzugehen. Dies ist bekannt dafür, dass Latenzspitzen

proportional zur Umlaufzeit zwischen Sender und Empfänger eingeführt werden. Auch windows basierte Staukontrolle - wie Reno, NewReno und CUBIC, die von den meisten TCP-Implementierungen verwendet werden - sind alle additiven AIMD-Algorithmen (additive increase/multiplicative decrease), deren Bandbreite bekanntlich stark von transienten Paketverlusten beeinflusst wird.

Harmony verwendet den RaptorQ Quellcode, um diese beiden Probleme zu bekämpfen. Jede Nachricht wird in Symbole kodiert, und Symbole werden über die Leitung gesendet, bis der Empfänger die erfolgreiche Dekodierung der Nachricht mit den empfangenen Symbolen bestätigt. Im Gegensatz zur Verwendung von Codes mit fester Rate wie Reed-Solomon, bei denen die Übertragung nach Erschöpfung der Symbole fehlschlägt, ermöglicht der Quellcode die unendliche, zeitnahe Erzeugung und Verwendung von Kodierungssymbolen.

6.4 Unterstützung für Home Nodes

P2P-Knoten (Nodes) in einem typischen Heimnetzwerk stellen ein großes, unverwechselbares Problem dar: Sie können von außen nicht erreicht werden, es sei denn, sie werden von ihrem privaten Internet-Router vermittelt, der eine Technik namens Network Address Translation (NAT) verwendet. Die Unterstützung für den eingehenden Datenverkehr durch diese Router ist unterschiedlich, und es wurden verschiedene Ansätze entwickelt, um verschiedene Arten von Routern zu umgehen. Insbesondere Router, die symmetrisches NAT implementieren, können nicht einfach umgangen werden, es sei denn, sie sind ausdrücklich so konfiguriert, dass sie andere Lochstanzmechanismen wie das Internet Gateway Device Protocol (IGDP) unterstützen.

Der P2P-Layer von Harmony versucht, den NAT-Mechanismus zu erkennen, hinter dem ein Knoten (Node) operiert, und verwendet den richtigen Workaround-Mechanismus, wie STUN, TURN, IGDP, etc. Insbesondere implementiert Harmony das gesamte Erkennungs- und Änderungsprotokoll ICE (Interactive Connectivity Establishment).

6.5 Unterstützung für Locator Mobility

Knoten (Nodes) können ihre IP-Adressen ändern, wobei einige Arten von Knoten (Nodes) mehr als andere. Ein solches Beispiel ist ein Laptop, der häufig zwischen verschiedenen Wi-Fi-Netzwerken wechseln kann, wobei sich seine IP-Adresse jedes Mal ändert. Wenn sich eine IP-Adresse eines Knotens (Nodes) ändert, werden alle bestehenden Transportverbindungen, die die IP-Adresse als lokalen oder entfernten Endpunkt verwenden, unterbrochen, und Anwendungen, die diese Transportverbindungen direkt nutzen, müssen Verbindungen mit der neuen IP-Adresse wiederherstellen, um fortzufahren. Ein solcher Verbindungsvorgang ist schwer zu implementieren und erfordert nur eine minimale Unterbrechung des Dienstes auf Anwendungsebene. Auch die Handhabung des Verbindungs Übergangs erschwert oft Anwendungsprotokolle auf Anwendungsebene (z.B. Basiskonsens-Protokolle).

Um dieses Problem zu lösen, führt der Netzwerk-Layer von Harmony mit dem Industriestandard Host Identity Protocol Version 2 (HIPv2) eine saubere Trennung zwischen Knoten Identität (kryptografisches Schlüsselpaar, das der Knoten (die Node) besitzt) und Knoten Ortung (Netzwerk-/Transport Layer-Lokalisierung, wo der Knoten (die Node) erreicht werden kann) ein. HIPv2 ermöglicht es Locatoren eines Knotens, sich im Laufe der Zeit zu ändern, während die Knoten Identität erhalten bleibt, indem es Mechanismen für die Lokalisierung von Locatoren, die Zuordnung von Node-zu-Node-Sicherheit und das Tunneln von Datenverkehr der obersten Ebene, der mit der lokalen /ferngesteuerten Knoten Identität als Endpunkte verbunden ist, bereitstellt.

7. Anreizmodell

7.1 Konsens-Belohnungen

Nach erfolgreicher anbindung eines Blocks wird eine protokoll mäßig festgelegte Anzahl neuer Token an alle Prüfer vergeben; die den Block im Verhältnis zu ihren Stimmanteilen unterzeichnet haben. Die Transaktionsgebühren werden den Prüfern in ähnlicher Weise vergütet.

7.2 Kürzung der Beteiligung

Für jedes Fehlverhalten, das vom Netzwerk erkannt wird, wird eine bestimmte Anzahl von gesetzten Token reduziert. Wenn ein Leiter beispielsweise den Konsensprozess nicht abgeschlossen und den Prozess des Leiterwechsels ausgelöst hat, werden die mit P_{vote} gesetzten Token gekürzt. Wenn Prüfer nachweislich einen unehrlichen Block unterschreiben, wird ihr ganzer Einsatz unter dem gleichen Shard vernichtet. Diese schwere Strafe soll jedes unehrliche Verhalten stark abschrecken und das Netzwerk so sicher wie möglich machen. Ein Beweis für Fehlverhalten können zwei unterschriebene Blöcke sein, die miteinander in Konflikt stehen. Jeder Prüfer kann eine Transaktion einreichen, um das Fehlverhalten eines anderen Prüfers nachzuweisen, und wenn überprüft wird, wird der gestrichene Token an den/die Prüfer belohnt.

7.3 Abhebung des Einsatzes

Fernangriffe

Proof-of-Stake-Blockchains neigen im Gegensatz zu Proof-of-Work-Blockchains dazu, unter Fern Angriffen zu leiden. Dies sind Angriffe, die die Tatsache nutzen, dass Nachweise auf Signaturen und nicht auf ressourcenintensiven Aufgaben basieren. Bei einem Fernangriff werden die privaten Schlüssel ehrlicher Prüfer lange nach ihrer Verwendung gestohlen, und der Angreifer ist in der Lage, eine gespaltene Blockchain zu erstellen, indem er gefälschte Blöcke mit diesen Schlüsseln signiert. In diesem Fall haben neue Prüfer, die dem Netzwerk beitreten, keine Möglichkeit, zwischen der ursprünglichen, legitimen Kette und der simulierten Kette des Angreifers zu unterscheiden.

Fernangriffe finden in den folgenden beiden Szenarien statt. Der private Schlüssel kann entweder durch einen Mangel an Sicherheit bei den Prüfern in entwendet werden, oder allgemeiner durch die Tatsache, dass ein Prüfer, nachdem er seine Token abgezogen hat, finanziell profitieren möchte, indem er einem Angreifer seinen privaten Schlüssel verkauft. Außerdem wird (per Design) jedem Satz von Prüfern vertraut, um den Transaktionsblock zu genehmigen, der auch den nächsten Satz von Prüfern bestimmt. Nachdem genügend privater Schlüssel (d.h. diejenigen, die zusammen mehr als $\frac{2}{3}$ der Stimmrechte an einem Shard halten) entwendet wurden, hat ein Angreifer die volle Kontrolle darüber, wer die nachfolgenden Prüfer sind.

Verteidigung der Fernangriffe: Resonante Kollegien

Die Proof-of-Work-Blockchain schützt vor den oben genannten Angriffen, indem es ehrlichen Prüfern eine objektive Methode des Forks bietet. In einer Proof-of-Work-Blockchain ist ein Fork zur Auswahl der rechtmäßigen Kette der akkumulierte Arbeitsaufwand in Form von berechneten Hashes.

In einer Proof-of-Stake-Blockkette ist das einzige objektive Maß, mit dem zwischen den Forks gewählt werden kann, die Gesamte Gewichtung der Signaturen, mit denen jeder Block genehmigt wird. Wenn wir diese gewichteten Signaturen verwenden, um zwei verschiedene Blöcke zu vergleichen, kommen wir zu der folgenden Gleichung, um zu bestimmen, wann eine Chain gespalten werden kann:

Sicherheit = "Blockfreigabe Schlüsselgewicht" - "Kompromittiertes Schlüsselgewicht”

Das "Gewicht des Blockgenehmigungsschlüssels" bezeichnet das Stimmrecht der Schlüssel die auf dem Block unterzeichnet sind. Wenn nach dem Gewicht des Einsatzes mehr private Schlüssel gefährdet sind, als zur Genehmigung eines Blocks verwendet wurden, kann der Block geforked werden. Bis dahin werden Prüfer immer die ursprüngliche, legitime Version des Blocks bevorzugen.

Harmony maximiert die Sicherheit jedes Blocks in seiner Proof-of-Stake-Blockchain, indem es diese Gleichung maximiert. Es ist nicht möglich, auslaufende private Schlüssel langfristig zu deaktivieren. Harmony motiviert die Prüfer stattdessen, das Genehmigungs Gewicht jedes Blocks zu maximieren, nachdem ein Quorum erreicht wurde. Dies geschieht durch die Verpflichtung der Prüfer, jeden vom Quorum genehmigten Block zu unterzeichnen, bevor sie diesen Prüfern erlauben, ihren Einsatz zurückzuziehen. Diese neuen zusätzlichen Signaturen müssen nur innerhalb der Blockchain existieren und nicht im Konsens Zeitpunkt für jeden Block generiert werden. Aus diesem Grund können die neuen Signaturen zu nachfolgenden Blöcken hinzugefügt werden, wenn die Prüfer beschließen, ihren Einsatz zurückzuziehen, und so können sie die Sicherheit der Chain frei erhöhen, ohne ihre Lebensdauer zu beeinträchtigen.

8. Zukünftige Forschung

8.1 Betrugsnachweise

Die Fähigkeit, das Fehlverhalten von Prüfern zu beweisen, ist wichtig, damit ein Light Client den empfangenen Blockdaten vertrauen kann. Bei der Kommunikation zwischen Shards ist jeder Shard ein Light Client für andere Shards. Die Gewährleistung, dass zwischen Shards gesendete Nachrichten vertrauenswürdig sind, ist für die Datenkonsistenz zwischen Shards von entscheidender Bedeutung. Wir forschen aktiv am Thema Datenverfügbarkeit [29] und Betrugsnachweise [2], um unser Protokoll zu sichern.

8.2 Status Entbundene Prüfer

In einer Hochdurchsatz-Blockchain wird die Größe der Blockchain-Daten schneller wachsen als bei bestehenden Chains, was für neue Prüfer ein großes Problem bei der schnellen Synchronisierung darstellt. Dies macht den Resharding-Prozess problematisch, denn wenn neue Prüfer nicht rechtzeitig synchronisiert werden können, kann die Prüffähigkeit der Prüfer nicht erreicht werden, damit ein neuer Block genehmigt werden kann, und selbst wenn die Prüffähigkeit erreicht wird, würde die Sicherheit des Protokolls beeinträchtigt. Die Bearbeitung von Statusblöcken ist eine Maßnahme zur Minderung des Problems, aber sie ist nicht optimal, da der Status mit der Chain immer weiter wächst. Wir arbeiten aktiv daran, einen “Statuslosen” Client [30,31] zu aktivieren, bei dem Prüfer nicht den gesamten Status der Blockchain synchronisieren müssen, um Transaktionen zu überprüfen.

Referenzen

[1] J.R. Douceur, The Sybil attack, in: 1st International Workshop on Peer-to-Peer Systems (IPTPS 02), 2002.
[2] Al-Bassam, M., Sonnino, A., & Buterin, V. (2018). Fraud Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities. CoRR, abs/1809.09044.
[3] Vasin, P. (2014) Blackcoin’s Proof-of-Stake Protocol v2, https://blackcoin.co/blackcoin-pos-protocolv2-whitepaper.pdf
[4] A. Kiayias, I. Konstantinou, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889, 2016. http://eprint.iacr.org/.
[5] P. Daian, R. Pass and E. Shi, Snow White: Robustly reconfigurable consensus and applications to provably secure proofs of stake, Cryptology ePrint Archive, Report 2016/919, 2017.
[6] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. https://eprint.iacr.org/2017/913.pdf.

[7] M. Zamani, M. Movahedi, and M. Raykova, “RapidChain: A Fast Blockchain Protocol via Full Sharding.” Cryptology ePrint Archive, Report 2018/460, 2018. https://eprint.iacr.org/2018/460.
[8] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “Omniledger: A secure, scale-out, decentralized ledger via sharding,” in 2018 IEEE Symposium on Security and Privacy (SP), pp. 19–34, 2018.
[9] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, pages 17–30, New York, NY, USA, 2016. ACM.
[10] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In 23rd Annual Network and Distributed System Security Symposium, NDSS, 2016.
[11] The QuarkChain Team. Cross Shard Transaction.
https://github.com/QuarkChain/pyquarkchain/wiki/Cross-Shard-Transaction
[12] The Zilliqa Team. The zilliqa technical whitepaper. https://docs.zilliqa.com/whitepaper.pdf, August 2017.
[13] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at https://bitcoin.org/ bitcoin.pdf.
[14] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI ’99), New Orleans, Louisiana, February 1999.
[15] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In Proceedings of the 25th USENIX Conference on Security Symposium, 2016.
[16] Drijvers, M., Edalatnejad, K., Ford, B., & Neven, G. (2018). Okamoto Beats Schnorr: On the Provable Security of Multi-Signatures. IACR Cryptology ePrint Archive, 2018, 417.
[17] B. Alangot M. Suresh A. S Raj R. K Pathinarupothi K. Achuthan "Reliable collective cosigning to scale blockchain with strong consistency" Proceedings of the Network and Distributed System Security Symposium (DISS‘18) 2018.
[18] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017.
[19] Baruch Awerbuch and Christian Scheideler. Towards a scalable and robust DHT. In Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA ’06, pages 318–327, New York, NY, USA, 2006. ACM.
[20] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In CRYPTO 2018, 2018.
[21] D. Boneh, B. B¨unz, and B. Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. https://eprint.iacr.org/2018/712.
[22] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In Proceedings of the 28th Annual Symposium on Foundations of Computer Science, SFCS ’87, pages 427–438, Washington, DC, USA, 1987. IEEE Computer Society.
[23] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping Authorities “Honest or Bust” with Decentralized Witness Cosigning. In 37th IEEE Symposium on Security and Privacy, May 2016.

[24] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. CoRR, abs/1710.09437, 2017.
[25] E. Syta, P. Jovanovic, E. Kokoris-Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable Bias-Resistant Distributed Randomness. In 38th IEEE Symposium on Security and Privacy, May 2017.
[26] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series consensus system, January 2018.
[27] The Ethereum Foundation. Ethereum Whitepaper.
https://github.com/ethereum/wiki/wiki/White-Paper.
[28] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, ASIACRYPT '01, pages 514–532, London, UK, UK, 2001. Springer-Verlag. https://www.iacr.org/archive/asiacrypt2001/22480516.pdf
[29] The Ethereum Team. A note on data availability and erasure coding.
https://github.com/ethereum/research/wiki/A-note-on-data-availability-and-erasure-coding
[30] A. Chepurnoy, C. Papamanthou, Y. Zhang. Edrax: A Cryptocurrency with Stateless Transaction Validation. Cryptology ePrint Archive, Report 2018/968.
[31] V. Buterin. The Stateless Client Concept.
https://ethresear.ch/t/the-stateless-client-concept/172
[32] Derek Leung, Adam Suhl, Yossi Gilad, and Nickolai Zeldovich. Vault: Fast bootstrapping for cryptocurrencies. Cryptology ePrint Archive, Report 2018/269, 2018.
[33] Ethereum Wiki. On Sharding Blockchain.
https://github.com/ethereum/wiki/wiki/Sharding-FAQs
[34] M. F. Nowlan, J. Faleiro, and B. Ford. Crux: Locality-preserving distributed systems. CoRR, abs/1405.0637, 2014.
[35] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In 23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016. The Internet Society, 2016.
[36] Paul Dworzanski. A note on committee random number generation, commit-reveal, and last-revealer attacks. http://paul.oemm.org/commit_reveal_subcommittees.pdf.
[37] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.
[38] David K. Gifford, Doctoral Dissertation, Information Storage in a Decentralized Computer System.
[39] Prince Mahajan, Lorenzo Alvisi, and Mike Dahlin Consistency, Availability, and Convergence , Technical Report (UTCS TR-11-22) http://www.cs.cornell.edu/lorenzo/papers/cac-tr.pdf
[40] Wyatt Lloyd et. al, Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS, Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP'11). https://www.cs.cmu.edu/~dga/papers/cops-sosp2011.pdf
[41] Peter Bailis†, Ali Ghodsi, Joseph M. Hellerstein, Ion Stoica, Bolt-on Causal Consistency, [SIGMOD'13].
[42] Postel, J. (1981). Transmission control protocol specification. *RFC 793.*

[43] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., & Minder, L. (2011). RaptorQ forward error correction scheme for object delivery (No. RFC 6330).