Challenges On The Road To Scalable Blockchains

Submitted as part of Honours Project (Semester 2)

Anurag Jain
Machine Learning Lab

Sujit Gujar
Machine Learning Lab

Abstract

Blockchain, one of the greatest innovation of the last decade, was invented to design a first ever decentralized cryptocurrecny *Bitcoin*. Thought it overcomes traditional impossibilities in decentralized systems, it faces many challenges, especially performance and fairness. In this semester, we studied some of the key research papers addressing these challenges. This report summarizes these papers briefly and list few challenges go ahead.

Contents

1

Introduction

1.1 Blockchain

A blockchain is a distributed database with the following unique properties make it suitable for preserving data among multiple nodes:

- Append-only
- Transparent
- Incorruptible
- Secure from adversaries
- Timestamped
- Consensus (Copies on all the nodes are consistent with each other)

1.2 Theme

This semester we explore generalizations of Nakomoto Consensus to solve the Security-Scalability Tradeoff, picking up from the previous semester. OHIE (Section 2) presents an approach by composing multiple parallel chains as instances of Nakomoto Consensus while SPECTRE (Section 3) and PHANTOM (Section 4) generalize the block-*chain* to block-*DAG* and then propose different rules to resolve conflict sets. We also study a few high level results pertaining to blockchain systems.

The report is divided into 2 Parts. In the first part, we discuss protocol related papers and in the second part we discuss papers presenting theoretical results about blockchain.

1.3 Part 1: Protocol-based papers

Researchers have proposed many protocols which could potentially offer substantial improvements over Bitcoin in terms of speed, security and efficiency. In this section we describe the summary of such papers proposing new protocols.

1.4 Part 2: Theoretical Results

In this section we summarise papers that presented novel high level ideas and results. The paper titled "Nonlinear Blockchain Scalability: a Game-Theoretic Perspective" (7) presents an important result that no blockchain system can achieve full verification, high scalability, and low finality-duration simultaneously. It provides a key insight to the relation between the complexity of verification and scalability. We also discuss a paper that presents security challenges in developing Proof-of-Stake protocols.

Part 1: Protocol-based Papers

2

OHIE: Blockchain Scaling Made Simple

[Read7]
Authors H. Yu, I. Nikolic, R. Hou, and P. Saxena. IEEE SNP 2020.

2.1 Solution Proposed

The paper proposes a novel permissionless blockchain protocol OHIE to scale the throughput of a blockchain while providing security guarantees (ref. Security-Scalability tradeoff), which explicitly aims for simplicity.

2.1.1 OHIE Protocol

OHIE composes k (e.g., $k = 1000$) *parallel* instances or so-called "chains" of Nakamoto consensus. Each chain has a distinct genesis block, and the chains have ids from 0 to $k-1$ (which can come from the lexicographic order of all the genesis blocks). For each chain, Bitcoin's longest-chain-rule is followed. The miners in OHIE extend the k chains concurrently, they are forced to evenly split their computational power across all chains.

Block Ordering OHIE uses the following algorithm to output a total order of blocks which is known as *sequence of confirmed blocks* (SCB).

Algorithm 1 Output SCB

```
for all $i$ in $0 \dots k-1$ do
    $W_i \leftarrow$ longest chain for the $i^{th}$ instance
    $P_i \leftarrow$ remove last $T$ blocks from $W_i$
    $l_i \leftarrow$ rank of last block in $W_i$
end for
confirm_bar $\leftarrow \min_{i \in 0 \dots k-1} l_i$
confirm_set $\leftarrow \phi$
for all $i$ in $0 \dots k-1$ do
    confirm_set $\leftarrow$ confirm_set$\cup${all blocks in $P_i$ with rank $\leq$ confirm_bar}
end for
sort blocks in confirm_set by rank (break tie by choosing smaller chain id)
return confirm_set
```

2.2 Security Guarantees

$$\begin{tikzcd}
\text{OHIE SCB} \arrow[d, "\text{as secure as}"] \\
\text{OHIE Individual Chain} \arrow[d, "\text{as secure as}"] \\
\text{Nakomoto}
\end{tikzcd}$$

Figure 1: Inheritance of security properties

The proof of safety and liveness properties of OHIE is given via a reduction from those of Nakomoto consensus.

Lemma 2.2.1. *Consider any given i where $0 \leq i \leq k-1$, and any given adversary $\mathcal{A}$ for (k, p, λ, T)-OHIE. There exists some adversary $\mathcal{A}_i'$ for (p, λ, T)-Nakamoto such that the following two random variables are strongly statistically close:*

- $\texttt{Chainview}_i(\texttt{EXEC}(\mathit{OHIE}, \texttt{k}, \texttt{p}, \lambda, \texttt{T}, \mathcal{A}))$
- $\sigma_i^\tau(\texttt{Chainview}(\texttt{EXEC}(\mathit{Nakamoto}, \texttt{p}, \lambda, \texttt{T}, \mathcal{A}_\texttt{i}')))$

Where the hash of a valid block needs to have $\log_2 \frac{1}{p}$ leading zeros, λ is the security parameter (i.e., the length of the hash output), and T is the number of blocks that we remove from the end of the chain in order to obtain partially-confirmed blocks. Here τ is the randomness in $\texttt{EXEC}(\mathit{Nakamoto}, \texttt{p}, \lambda, \texttt{T}, \mathcal{A}_\texttt{i}')$. The random variable $\sigma_i^\tau(\texttt{Chainview}())$ is the same as $\texttt{Chainview}()$, except that we replace each block B' in $\texttt{Chainview}()$ with another block $\sigma_i^\tau(B')$. The mapping $\sigma_i^\tau()$ is some one-to-one mapping from each block B' among all the blocks on all the nodes in (p, λ, T)-Nakamoto to some block $\sigma_i^\tau(B')$ on all the nodes in (k, p, λ, T)-OHIE. The mapping $\sigma_i^\tau()$ guarantees that i) B' is an honest block iff $\sigma_i^\tau(B')$ is an honest block, and ii) B' extends from A' iff $\sigma_i^\tau(B')$ extends from $\sigma_i^\tau(A')$.

The lemma 2.2.1 claims that every single chain of OHIE is atleast as secure as the Nakomoto Chain in Bitcoin. Thus, they follow the persistence and liveness properties inherited from Bitcoin. The authors then show that if each chain follows the persistence and liveness properties then by the virtue of the SCB algorithm, the SCB would also follow persistence and liveness properties.

2.3 Comparative Study

Comparison with Bitcoin As per the claim of the authors, the above protocol is equivalent to Bitcoin if $k = 1$ (only one chain is used) otherwise it offers k times the throughput of Bitcoin while providing the same security. What suffers however is the time to finality, which is claimed to increase with $O(\log k)$. This should not degrade the performance of the system since it is practically determined by a parameter ϵ, such that the probability of reverting the transaction becomes greater than $1 - \epsilon$ and time to finality grows with $O(\log \frac{1}{\epsilon})$

2.3.1 Comparison with other protocols

The main advantage of OHIE over other protocols which also offer increased throughput is the formal end-to-end security proof provided by the authors. This should ensure that there are no major security flaws which could not be discovered via informal arguments. It could also offer better decentralization (20 times more) in comparison to GHOST protocol and Bitcoin-NG.

Spectre confirms block without guaranteeing a block order, but OHIE guarantees a block order which makes it suitable for systems like Ethereum, where a total order over transactions is required.

2.4 Related Work

Chainweb is a blockchain protocol which uses multiple chains and blocks include merkle proofs over multiple chains [8] similar to OHIE.

2.5 Novel Ideas

The paper presents a scalable blockchain protocol with a formal security proof by composing together multiple instances of Nakomoto Consensus. The benefit of having an end-to-end security proof is to ensure that there are no possible attacks.

3

SPECTRE: A Fast and Scalable Cryptocurrency Protocol

[Read5]
Authors Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Cryptology ePrint Archive 2016.

3.1 Solution Proposed

The paper proposes a generalization of blockchains into block-DAGs (Directed Acyclic Graphs) and a voting algorithm to determine the subset of accepted blocks which they claim could alleviate the security-scalability tradeoff imposed by the Nakomoto Consensus. However this weakens the "Liveness" property since in order for a transaction to be *robustly* accepted by any node, no conflicting transactions must be published for a short while after publishing the transaction. In practice, this is not an issue since two conflicting payments that are published concurrently could only have been created by a dishonest user, hence we can afford to delay the acceptance of such transactions without harming the usability of the system.

3.2 Protocol

3.2.1 Mining Protocol

1. When creating or receiving a block, transmit the block to all peers.
2. When creating a block, embed in its header a list containing the hash of all leaf-blocks (blocks with in-degree 0) in the locally-observed DAG.

3.2.2 Voting Rule

For comparing conflicting blocks, the protocol generates a pairwise ordering of blocks via the described voting algorithm and the block which comes before is selected from the conflict set[a]. Consider the set of blocks which vote for a block x over a conflicting block y.

- x itself
- All the blocks from which only x is reachable. (Future blocks which have seen x) *They claim x came before y*
- For blocks from which both x and y is reachable, recursively apply the voting algorithm on the subset of blocks reachable from the current block and vote for the majority block determined recursively. *They only amplify the original claim*
- For blocks from which neither x nor y is reachable i.e., they don't know either of the blocks since they were mined before x and y have the same vote as the majority of the blocks from which they are reachable. *Avoids a pre-mining attack and benefits the miners who include as many references to leaves of the DAG as possible.*

[a]Note that the voting here is implicit in the sense the blocks do not store the votes explicitly but their votes are determined via the stucture of the DAG

3.2.3 Acceptance of Transaction

A transaction is said to be accepted iff:

1. all of its inputs have been accepted
2. all conflicting transactions that precede it, have been rejected
3. all conflicting transactions which are reachable from the block are preceded by the block containing the transaction i.e., the conflicting transactions are guaranteed to be rejected.

The paper also describes a *robust* acceptance rule which guarantees that the transaction will persist with high probability.

Evaluating Risk The authors propose a method to allow users to measure the robustness of accepted transactions based on the parameters observed and set by the user. The function Risk(G, x, y) provides the upper bound on the probability that an attacker will be able to reverse the relation $x \prec y$ (y could be a specific block or an unseen block withheld by an attacker or yet to be created). It is evaluated by assuming the attacker behaves *optimally* to calculate the probability of the attacker winning any of the four races:

- $f_{\text{pre\_mine}}$ - Upper bound of the probability that the attacker has won the race during the pre-mining phase.
- $f_{\text{pre\_pub}}$ - Upper bound of the probability that some honest blocks $\in \text{anticone}(x, G)$ because x might have been withheld by the attacker.

3.3 Proof of Security Properties

Since SPECTRE is a DAG-based blockchain, we cannot use the same properties defined in the Bitcoin Backbone Protocol [Read3] since they were applicable only to linear blockchains. Instead we have properties which are defined differently, but they have the same essence.

1. **Weak Liveness** - It says that some honest node will eventually *robustly* accept a block. Formally, Let t be the current time, and assume that $x \in G_t^{\text{pub}}$. Let ψ be the first time s at which for some honest node v: $\text{Risk}(x, y, G_s^v) < \epsilon$. Then, conditioned on the event where $y \notin G^{\text{pub}}$, the expectation of $\psi - t$ is finite.
   Note that it is said to be weaker than the Liveness property of Bitcoin, this is because the liveness property is conditioned on the fact that no conflicting block is published before some honest node accepts a block.
2. **Safety** - It says that once some honest node has *robustly* accepted a block, eventually every honest node will *robustly* accept it. Formally, For any $v \in \textit{honest}$, if $\text{Risk}(x, y, G_t^v) < \epsilon$ then, with probability of $1-\epsilon$ at least, there exists a τ such that $\forall u \in \text{honest}$, $\forall s \geq \tau : \text{Risk}(x, y, G_s^u) < \epsilon$, and $\mathbb{E}[\tau - t] < \infty$.
3. **Progress** - It says that once a block is accepted for some ϵ risk, it will eventually accepted for all $\epsilon' < \epsilon$ risk as well. This means that the risk associated with a block can only decrease. Formally, For any $v \in \textit{honest}$, if $\text{Risk}(x, y, G_t^v) < \epsilon$ then, with probability of $1-\epsilon$ at least, for any ϵ' there exists a ϕ such that, $\forall s \geq \phi : \text{Risk}(x, y, G_s^u) < \epsilon$, and $\mathbb{E}[\phi - t] < \infty$.

3.4 Comparative Study

3.4.1 Comparison with Bitcoin

One of the key achievement's of SPECTRE is its ability to scale without compromising the security threshold. The security threshold of the Bitcoin protocol goes to zero as $D\lambda$ increases whereas in SPECTRE it stays at 50%.

3.4.2 Comparison with GHOST

GHOST was able to modify Bitcoin's longest chain rule in order to maintain the security threshold at 50% regardless of the throughput but it can be shown that it is vulnerable to be attacks on the Liveness property [6] but it is not the case with SPECTRE.

3.5 Related Work

There has been significant research attempting to solve the Security-Scalability tradeoff, GHOST can be considered as an intermediate to SPECTRE from Bitcoin by including off-chain blocks into the consensus protocol but SPECTRE takes it to the next level by including the off-chain blocks into the ledger itself.

3.6 Conclusion

SPECTRE is a new cryptocurrency protocol that explores the paradigm of blockchain as a voting mechanism but solves a weaker problem than traditional consensus protocol (due to *Weak* Liveness). The key to its success is due to its ability to delay the decision between visibly double-spent transactions while not delaying virtuous transactions.

3.7 Novel Ideas

The paper presents a protocol that generalization of blockchain to block-DAG. It considers that each node participates in consensus by implicitly voting via blocks and generalizes Nakomoto's voting rule to block-DAG.

4 PHANTOM and GHOSTDAG

[Read6]
Authors Yonatan Sompolinsky, Shai Wyborski and Aviv Zohar. Cryptology ePrint Archive 2018
PHANTOM and GHOSTDAG are yet another generalizations of Nakomoto Consensus to Block-DAGs.

4.1 PHANTOM

PHANTOM solves an optimization problem over the blockDAG distinguish between blocks mined properly by honest nodes and those created by non-cooperating nodes who chose to deviate from the mining protocol, let this set be called Blue and the rest of the blocks called Red. Once the set is identified then, for any block $B \in \text{Blue}$ chosen in topological order, all blocks in $\textit{past}(B) \cap \text{Red}$ are placed after B in the same topological order.

4.1.1 Maximum k-cluster SubDAG

Just like Bitcoin, PHANTOM relies on the ability of honest nodes to communicate to their peers recent blocks in a timely manner (a dishonest node trying to double spend will hide his malicious block from the honest network until the original block gets accepted). Let λ be the expected block creation rate of the network and D be the network diameter, then for a block mined at time t the expected number of blocks in its anticone set will contain expected $2\lambda D$ blocks (It won't contain pointers to blocks mined in the time period from $[t-D, t]$ and blocks mined in the time period from $[t, t+D]$ won't be point to it. Thus, blocks mined between $[t-D, t+D]$ will not be topologically related to it). Since this is a poisson process, we can safely say that the number of blocks that can be mined in the interval of $2D$ will be bounded by some k with high probability. Therefore the set of blocks created by honest nodes can be said to be "well-connected". Formally,

k-Cluster Given a DAG $G = (C, E)$, a subset $S \subseteq C$ is called a k-cluster, if $\forall B \in S : |anticone(B) \cap S| \leq k$.

PHANTOM utilizes this fact and selects the largest well-connected set within the DAG (which it calls Blue) by solving the following optimization problem:

$$S^* = \max_S \; S = \{B : |anticone(B) \cap S| \leq k\}$$

Unfortunately, this problem is $\mathbb{NP}$-Hard to solve [4], which makes it unsuitable for practical applications. GHOSTDAG tries to use a greedy algorithm with a similar intuition to find a sub-optimal solution.

4.2 GHOSTDAG

GHOSTDAG finds a k-cluster using a greedy algorithm. The algorithm makes the following assumptions which not hold always but it doesn't matter since we are aiming for a sub-optimal solution:

- **Greedy Choice Property** - Pick the block giving the largest $|S|$ and include the blocks in its anticone which maintain the k-cluster property in the order of the topological sort.
- **Optimal Substructure** -

$$\mathrm{MCS}_k(G) \cap G' = \mathrm{MCS}_k(G') \; \forall G' \subseteq G$$

The algorithm presented in the paper is $O(n^2)$ which is still not suitable for practical applications, however the authors have also shared an efficient implementation [11].

4.3 Security Properties

Given a block creation rate $\lambda > 0, \delta > 0$, and $D_{\max} > 0$, if $D_{\max}$ is equal to or greater than the network's propagation delay diameter D, then the security threshold of GHOSTDAG, parameterized with $k(D_{\max}, \delta)$, is lower bounded by $\frac{1}{2}(1-\delta)$. Where δ is a parameter such that

$$\Pr[\text{Size of any Anticone} > k] < \delta$$

Therefore, one can say that unlike Nakomoto consensus, GHOSTDAG scales. Since the block creation rate follows a Poisson process, for an arbitrary block B created at time t, $k(D_{\max}, \delta)$ bounds the number of additional blocks created in the time interval $[t-D_{\max}, t+D_{\max}]$.

$$\begin{aligned} k\left(D_{\max}, \delta\right) &:= \min\left\{\hat{k} \in \mathbb{N} : f\left(\hat{k}, D_{\max}\right) < \delta\right\} \\ f\left(\hat{k}, D_{\max}\right) &:= \max\left\{\sum_{j=\hat{k}+1}^{\infty} e^{-2\cdot C} \cdot \frac{(2\cdot C)^j}{j!}, \frac{2\cdot C}{\hat{k}+2\cdot C}\right\} \end{aligned} \tag{1}$$

where C is defined as $\lambda D_{\max}$.

4.3.1 Confirmation Times

One can observe that the anticone of a given block closes quickly, using this one can measure the "risk" associated with accepting a block. The function Risk_u is defined by the probability that from the point of view of node u a block that did not precede B in time $t+r$ will later come to precede it:

$$\mathrm{Risk}_u(B, t, r) := \Pr(\mathcal{E})$$

where $\mathcal{E} = \{\exists s > t+r, \exists C \in G_s^u : B <_{G_{t+r}^u} C \wedge C <_{G_s^u} B\}$.

A block is said to be accepted by a node u if $\mathrm{Risk}_u \leq \epsilon$ and ideally we want it to vanish exponentially fast so as to ensure speedy confirmation times.

4.4 Related Work

The GHOSTDAG algorithm can be viewed as a generalization of the GHOST Algorithm proposed for blockchains to block-DAGs(this was studied by me in the previous semester [10]). This work is most similar to the SPECTRE protocol (3). SPECTRE enjoys both high throughput and fast confirmation times. It uses the structure of the DAG as representing an abstract vote regarding the order between each pair of blocks. One caveat of SPECTRE is that the output of this pairwise ordering may not be extendable to a total order, due to possible Condorcet cycles. Another related caveat is that SPECTRE does not guarantee convergence of the ordering between two blocks that were published in time proximity to one another. This weak liveness property is showed to suffice for the use case of payments, where conflicts in two such blocks can only harm a malicious user that published a double spend. PHANTOM solves these issue and provides a linear ordering over the blocks of the DAG. As such, PHANTOM can support consensus regarding any general computation, including general Smart Contracts, which SPECTRE cannot handle. However, PHANTOM has much slower confirmation times as compared to SPECTRE.

4.5 Novel Ideas

The paper picks up from SPECTRE to present a protocol that produces total block ordering using similar ideas.

5

The Bitcoin Backbone Protocol

[Read3]
Authors Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. EuroCrypt 2015.

5.1 Problems Addressed

The paper analyzes the consensus mechanism used in Bitcoin dubbed as the "Bitcoin Backbone Protocol". The paper assumes a "flat-model" and proves two of the mechanism's fundamental properties, "chain quality" and "common prefix".

The paper also analyzes Nakomoto's proposal for solving the Byzantine Agreement problem and shows that it fails to satisfy the *Validity* and *Adversity* correctness conditions. The authors propose an alternative based on the *Bitcoin Backbone Protocol* that solves the problem.

5.2 Analysis

5.2.1 q-bounded Synchronous Model

For the sake of analysis, we assume that the execution takes place synchronously in rounds, where in each round all parties can communicate with each other.

Parties are allowed to perform at most q attempts at Proof-of-Work per round. Parties with higher computational power could be modelled as a coalition of many parties.

There are n parties, out of which t are controlled by the adversary. It is assumed that the adversary has the ability to communicate between rounds while the honest nodes don't.

Notation Consider three random variables associated with the i^{th} round,

$$X_i = \begin{cases} 1 & \text{if \textit{atleast} one honest party obtains} \\ & \text{a POW in the round} \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_i = \begin{cases} 1 & \text{if \textit{exactly} one honest party obtains} \\ & \text{a POW in the round} \\ 0 & \text{otherwise.} \end{cases}$$

$$Z_i = n \quad \text{if the adversary finds } n \text{ blocks in the round}$$

Also, $\mathbf{E}(X) = f$ for convenience.

(ϵ, λ) - Typical Execution An execution of λ consecutive rounds is said to be (ϵ, λ) - typical if X_i, Y_i and Z_i are close enough to their expected values for all rounds, i.e.

Let S be the set of λ consecutive rounds, such that $\lambda \geq 2/f$

$$(1-\epsilon)\mathbf{E}(X(S)) < X(S) < (1-\epsilon)\mathbf{E}(X(S)) \qquad \forall \quad i \in S$$
$$(1-\epsilon)\mathbf{E}(Y(S)) < Y(S) \qquad \forall \quad i \in S$$
$$Z(S) < \mathbf{E}(Z(S)) + \epsilon\mathbf{E}(X(S)) \qquad \forall \quad i \in S$$

It can be shown that an execution will be (ϵ, λ) - typical with probability $1 - e^{-\Omega(\epsilon^2 \lambda f)}$ (using Chernoff Bounds).

Honest Majority Assumption The adversary can control at most t out of n parties such that $t \leq (1 - \delta)(n-t)$ where $3(f+\epsilon) < \delta$ for some $\delta \in [0,1]$

5.3 Bounds on the probability of PoW

Let p be the probability of success of a single query. Then, for q queries,

$$\mathbf{E}(X) = f = 1 - (1-p)^{q(n-t)} \qquad (2)$$
$$\implies (1-f)pq(n-t) \leq f \leq pq(n-t) \qquad (3)$$
$$\implies pq(n-t)(1-p)^{q(n-t)} \leq f \leq pq(n-t) \qquad (4)$$

5.4 Desired Properties

5.4.1 Common Prefix

If C_1 and C_2 are the chains adopted by any two honest nodes P_1 and P_2 at rounds r_1 and r_2 respectively, then pruning the k blocks from the end of C_1 will result in a chain that is a valid prefix of C_2.

5.4.2 Chain Quality

The chain quality property Q_{cq}, parameterized by $\mu \in [0,1]$ and l states that for a chain C adopted by any honest party for any set of l consecutive blocks of the chain should contain at least μl blocks contributed by honest nodes.

Ideally, $1 - \mu$ (or the contribution of blocks by an adversary) should be directly proportional to the hashing power of adversarial nodes. Any scenario that violates this could be considered an attack on the blockchain.

5.4.3 Chain Growth

The chain growth property Q_{cg}, parameterized by $\tau \in [0,1]$ and s states that for a chain C adopted by any honest party should be extended by atleast τs blocks in any s consecutive rounds.

These three properties enable the two requirements for any robust transaction ledger:

- **Persistence**: Persistence states that once a transaction goes more than k blocks "deep" into the blockchain of one honest player, then it will be included in every honest player's blockchain with overwhelming probability, and it will be assigned a permanent position in the ledger (essentially, this means that all honest players agree on all the transactions that took place and in what order). In a more concrete Bitcoin-like setting, Persistence is essential to ensure that credits are final and that they happened at a certain "time" in the system's timeline (which is implicitly defined by the ledger itself).

- **Liveness**: The Liveness property states that as long as a transaction comes from an honest account holder and is provided by the environment to all honest players, then it will be inserted into the honest players' ledgers (assuming the environment keeps providing it as an input for a sufficient number of rounds) i.e, all transactions originating from honest account holders will eventually end up at a depth more than k blocks in an honest player's blockchain, and hence the adversary cannot perform a selective denial of service attack against honest account holders. Liveness implies Validity, because assuming the ledger is maintained for long enough, a majority of transactions originating from the honest parties will be included (despite the fact that honest parties may control a minority of blocks in the blockchain).

5.5 Bitcoin Backbone Protocol in the Bounded Delay Model

Consider the Bitcoin Backbone Protocol in a network where the synchronization (message passing between honest nodes) does not happen immediately but instead happens after Δ-rounds. We define new random variables (parallel to the ones defined in the original model).

$$X'_i = \begin{cases} 1 & \text{if } i \text{ is a } \Delta\text{-isolated successful round} \\ 0 & \text{otherwise.} \end{cases}$$

$$Y'_i = \begin{cases} 1 & \text{if } i \text{ is a } \Delta\text{-isolated uniquely successful round} \\ 0 & \text{otherwise.} \end{cases}$$

The i^{th} round is said to be isolated if no party managed to be successful in a window of Δ-rounds before and after the i^{th} round. We have

$$\mathbb{E}[X'_i] = f(1-f)^{\Delta-1} \geq f[1-(\Delta-1)f]$$

$$\mathbb{E}[Y'_i] \geq f(1-f)^{2\Delta-1} \geq f[1-(2\Delta-1)f]$$

Note that the probability of Δ-isolated uniquely successful rounds decreases which weakens the Honest Majority Assumption (we can tolerate lesser fraction of the computing power controlled by the adversary)

Typical Execution An execution is $(\epsilon, \lambda, \Delta)$ -typical, with $\epsilon \in (0,1), \lambda \geq 2/f$, and integer Δ, if, for any set S of at least λ consecutive rounds, the following hold.

- $(1-\epsilon)\mathbb{E}[X'(S)] < X'(S), \quad X(S) < (1+\epsilon)\mathbb{E}[X(S)]$, and $(1-\epsilon)\mathbb{E}[Y'(S)] < Y'(S)$
- $Z(S) < \mathbb{E}[Z(S)] + \epsilon\mathbb{E}[X'(S)]$
- No insertions, no copies, and no predictions occured

An execution is typical with probability $1 - e^{-\Omega\left(\epsilon^2 f^2 (1-f)^{4\Delta-2}\lambda\right)}$. (Reduced from the standard model) The bounds for chain quality, common prefix, and chain growth property are also weakened. This could be possibly offset by increasing k (the number of blocks dropped from the end) which would increase the time to finality. In case of a typical execution (with high probability) the following properties will hold (weaker than the original model)

- **Chain Growth** $\tau = (1-\epsilon)f(1-f)^{\Delta-1}$ instead of $(1-\epsilon)f$
- **Chain Quality** $\mu = 1 - \frac{1}{(1-\epsilon)(1-f)^{\Delta}} \cdot \frac{t}{n-t} - \frac{\epsilon}{1-\epsilon} \cdot \left(1+\frac{\Delta}{\lambda}\right)$ instead of $1 - \left(1+\frac{\delta}{2}\right) \cdot \frac{t}{n-t} - \frac{\epsilon}{1-\epsilon} > 1 - \left(1+\frac{\delta}{2}\right) \cdot \frac{t}{n-t} - \frac{\delta}{2}$
- **Common Prefix** $k \geq 2\lambda f + 2\Delta$ instead of $k \geq 2\lambda f$

5.6 Novel Ideas

The paper presents formal proofs of Nakomoto's proposal of Bitcoin and proposes important properties, *Liveness* and *Persistence* for blockchain protocols to meet in order to create a secure distributed ledger. The paper also analyzes the same protocol and proves the same properties in case of a Bounded Delay Model.

6

FruitChains: A Fair Blockchain

[Read4]
Authors Rafael Pass and Elaine Shi. PODC 2017.

6.1 Problems Addressed

Bitcoin is vulnerable to selfish mining [3] wherein the miners can collude to gain up to twice their *fair* share of rewards and transaction fees. *Fairness* refers to the property that the miners controlling ϕ fraction of the computing power gain ϕ fraction of the total reward. Due to large variance and low probability of a block being mined by a solo miner, the miners often form coalitions in the form of mining pools. This causes the blockchain to lose its decentralized nature. In Bitcoin, the transaction fees is expected to contribute majorly towards the miner's reward in the future, however this could contribute to instability since miners will be incentivized to create a “fork” and attempt to confirm the transaction themselves.

6.2 Solution Proposed

The authors introduce a δ-approximate fair (w.r.t. ρ attackers) blockchain protocol and reducing the block (i.e., a fruit) difficulty so that the miners are rewarded more often which reduces the need for joining mining pools.

6.3 FruitChain Protocol

FruitChain differs from Nakamoto’s blockchain protocol. Compare to Bitcoin blockchain, instead of putting a set of messages m into the block, authors come up with idea to put them into the “fruits” denoted as f.

$$\begin{tikzcd}
\text{Blocks} \\
\text{Fruits} \arrow[u, "\text{confirmed by}"'] \\
\text{Transactions} \arrow[u, "\text{confirmed by}"']
\end{tikzcd}$$

Figure 2: Flow of Confirmation

6.3.1 Fruits and Blocks

Fruits are similar to blocks but fruit requires additional requirements to be “hanged”.

Property	**Fruit**	**Block**
h_{-1}	Doesn’t Matter	Pointer to the previous block
h'	Pointer to the block from which the fruit hangs	Doesn’t Matter
η	Random Nonce	Random Nonce
m	Transaction Records	Doesn’t Matter
F	Doesn’t Matter	The Fruit Set
digest	Doesn’t Matter	Digest of the Fruit Set

Table 1: Piggybacked Parameters of Fruits and Blocks

Fruit A Fruit is a structure that stores the records of transaction. A Fruit is said to be valid iff: (i) The last κ bits of the hash h is less than the difficulty D_{p_f} (ii) It is included in a block j such that the block i from which it hangs from is recent, i.e., $j \geq i - R\kappa$ where R is the recency factor

Block A Block is the structure that includes multiple fruit inside of it. A block is said to be valid iff: (i) digest $= d(F)$ where d is a collision-resistant hash function (ii) F is a valid fruit-set (iii) The first κ bits of the hash h is less than the difficulty D_{p_1}

Mining Protocol:

1. Upon receiving a valid fruit:

$$F := F \cup \text{fruit}$$

2. Pick the longest chain and formulate the set $(h_{-1}; h'; \eta; \mathrm{d}(F'); \mathrm{m})$

3. Pick a random nonce η

   (a) if the last κ bits of the hash h is less than the difficulty D_{p_f}:
   Successfully mined a fruit; Broadcast fruit

   (b) if the first κ bits of the hash h is less than the difficulty D_{p_1}
   Successfully mined a block; Broadcast block

Block Ordering:

1. First extract a sequence of distinct fruits from chain, where if the same fruit is included multiple times, only the first occurrence is included. The extracted fruits are ordered by the first block that contains the fruit; and for fruits in the same block, follow the order in which the fruits are serialized within the block.

2. Output the sequence of records m contained in the extracted sequence of fruits, where records contained in earlier fruits are extracted earlier.

6.4 Notion of Fairness

A blockchain protocol is δ approximately fair w.r.t. ρ attackers if, with overwhelming probability, any honest subset controlling ϕ fraction of the compute power is guaranteed to get at least a $(1-\delta)\phi$ fraction of the blocks in every sufficiently long window $(\Omega(k/\delta))$, even in the presence of an adversary controlling a ρ fraction of the computation power. [1]

6.5 Security Properties

The security properties of the Fruitchain protocol is proved via the following theorem:

Theorem 6.1. *For any constant $0 < \delta < 1$, and any p, p_f, let $R = 17$ $\kappa_f = 2qR\kappa$, and $T_0 = 5\frac{\kappa_f}{\delta}$ Then the FruitChain protocol denoted Π_{fruit} (p, p_f, R) satisfies*

1. *κ_f- consistency (equivalent to the persistence property)*
2. *chain growth rate (T_0, g_0, g_1) where*
$$g_0 = (1-\delta)(1-\rho)np_f$$
$$g_1 = (1+\delta)np_f$$
3. *fairness (T_0, δ)*

The chain growth and fairness properties ensure the liveness property and consistency ensures persistence property.

6.6 Related Work

The piggybacked mining scheme is quite similar to the Byzantine Agreement Protocol based on the Bitcoin Backbone Protocol presented in [Read3]. The selfish mining problem was presented in [3].

6.7 Novel Ideas

The paper proposes a blockchain protocol to solve the Selfish Mining attack and improves decentralization of blockchain by reducing the incentive to form mining pools. The authors also put forward a new notion of δ-approximate fairness.

PART 2: PAPERS PRESENTING INTERESTING THEORETICAL RESULTS

7

NONLINEAR BLOCKCHAIN SCALABILITY: A GAME-THEORETIC PERSPECTIVE

[Read2]
Authors Lin Chen, Lei Xu, Zhimin Gao, Keshav Kasichainula, and Weidong Shi. Arxiv 2020.

The paper provides a formal framework to measure scalability and proof that no blockchain system can achieve full verification, high scalability, and low finality-duration simultaneously.

[1]Observe that δ-approximately fair $\iff$ Chain Quality $\mu \geq (1-\delta)(1-\rho)$

7.1 Framework

The formal framework provides the following metrics for a blockchain system.

7.1.1 Full Verification

A blockchain system satisfies the property of full verification if every block is verified by all the nodes in the system before it is finalized.

Sharding Many approaches towards increasing scalability try to bypass this property, for instance a sharding scheme that allows transactions to be processed by a subgroup of nodes (a sharding committee). A sharding scheme usually has a critical issue in terms of resilience, as the correctness of each transaction now solely depends on a subgroup of voters.

Any divide and conquer based solution would inevitably reduce the total number of verifications received by a transaction and such solutions will not satisfy the *full verification* property.

7.1.2 Scalability

The throughput of a blockchain system is the number of blocks n_b that can be added to the system in a fixed time. A blockchain system scales with the number of nodes m in the system if $n_b \to \infty$ when $m \to \infty$. Particularly, a blockchain system fully scales with the number of nodes m if $n_b = \Omega(m)$. A system like Bitcoin is not scalable since the number of blocks that can be added to a system does not increase with an increase in the number of nodes.

7.1.3 Finality Duration

The finality-duration of a blockchain system is the time difference between the time point when a block is appended and the time point when a block receives full verification. The finality-duration is considered low enough if it is independent of the number of nodes in the system. Systems like Bitcoin achieve low finality duration since block times are fixed at 10 minutes and after a fixed number of blocks are appended to a block, it is considered to be confirmed.

7.2 Impossibility Result

Theorem 7.1. *There does not exist a blockchain system that simultaneously satisfies (i) scalability; (ii) low finality-duration; and (iii) full verification.*

Proof. The proof by contradiction follows from a simple counting argument,
Let us assume that there exists such a blockchain system which achieves (i), (ii) and (iii). Then by definition, every block or transaction will receive verifications from all the nodes within a constant delay.
Let c_0 be the constant delay.

Consider an arbitrary node x and let τ_x be the fixed time it takes for node x to perform one verification.
Let the throughput of the blockchain be n_b, then by definition of scalability we have $n_b = n_b(m) \to \infty$ when $m \to \infty$.
Note that all the n_b blocks generated shall be verified by every node within the delay of c_0, which means every node should perform n_b verifications within c_0.
However, node x can only perform c_0/τ_x verifications, which is a constant. Since $n_b \to \infty$, when m is sufficiently large, $n_b > c_0/\tau_x$. Therefore, it is impossible for an arbitrary node x to complete all the verifications. $\square$

7.3 Related Work

This paper presents a high level result that encompasses wide literature on scaling blockchains. The nonlinear blockchain model used was presented in [9].

7.4 Novel Ideas

The paper presents an impossibility result in scaling blockchain protocols whilst mantaining low finality-duration and full verification based on an argument on computational complexity of verifying all the blocks. The notion of scalability used here is in terms of nodes participating in the system (which differs from other protocols presented earlier).

8

Formal Barriers to Longest-Chain Proof-of-Stake Protocols

[Read1]
Authors Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg.
ACM EC 2019.

The paper presents a model for Longest-Chain Proof-of-Stake Protocols in which they show the possibility of two attacks based on incentive-driven deviations. The "barrier" to these protocols is the fact that both of these attacks cannot be mitigated simultaneously.

8.1 Model

The model considered is an ideal network where every node has perfectly synchronized clocks, and every message is received with zero latency by every other node. As compared to the practical application, this model contains minimal assumptions. A Proof-of-Stake protocol must satisfy the following properties:

- **Chain Dependence** Block B's validity at time t only depends on B, t and B's predecessors. This implies that a block cannot conflict with a future block in the same chain.

- **Monotonicity** If a block B is valid at time t for a given graph $T(B)$, then it is valid for all graphs T' that contain $T(B)$ as a subgraph and for all times $t' \geq t$. This implies that once a transaction is accepted, it cannot be reversed by a future block in the same chain.

- **Validator** V_P takes as input a block B and outputs whether it is valid or not. V_P must be efficiently computable by every participant in the protocol. A block B is said to be *valid* at time t if and only if predecessor(B) is valid and upto this block, the owner of the coin c_B (the chosen coin) is the miner(B) and the timestamp of the block is between the timestamp of its predecessor and the current time.

- **Mining Function** M_P takes as input a block A, a coin c and timestamp t and outputs a block, the key difference from the Proof-of-Work protocols being that it is efficiently computable here. One should note that $M_P(A, c, t)$ is only efficiently computable by the owner of the coin c.

A longest chain Proof-of-Stake protocol satisfies an additional property:

1. **Longest Chain Selection** As a block is added to a chain, its "*score*" will increase monotonically and an honest user will mine only on the chain with the highest score.

8.2 Properties

The paper highlights the two desirable but complementary properties in longest chain Proof-of-Stake protocols

8.2.1 (Un)-Predictability

A protocol is said to be unpredictable if the miners are not able learn that they are eligible to mine a block until shortly before it is due to be mined. If this is not the case, then the attacks such as double-spending and selfish mining can be potentially more profitable.

D-locally predictable A coin c is D-locally predictable at block A for timestamp t if owner(c) can efficiently predict whether or not there will exist a block B with $c_B = c$ such that $V_P(B) = 1$, where predecessor$^D(B) = A$ and $t_B = t$. (One can observe that for any Proof-of-Stake protocol, every coin c is 1-locally predictable at every block A for every timestamp $t > t_A$.)

D-globally predictable A coin c is D-globally predictable at block A for timestamp t if **every** participant (not only the owner) of the protocol can efficiently predict whether or not there will exist a block B with $c_B = c$ such that $V_P(B) = 1$, where $\text{predecessor}^D(B) = A$ and $t_B = t$. (This should not happen if $M_P(A, c, t)$ is only efficiently computable by the owner of the coin c.)

8.2.2 (Non)-Recency

This property is complementary to local predictability. This requires that the eligibility to mine a block depends on the recent history. (Note that For any D, any block A and any timestamp t, a coin c is either D-locally predictable or D-recent.)

D-recent A coin c is D-recent at a block A for timestamp t if the owner of c *cannot* efficiently predict whether or not there will exist a block B such that $V_P(B) = 1$, where $\text{predecessor}^D(B) = A$, $c_B = c$, and $t_B = t$.

8.3 Attacks

For both these properties, there are incentive driven attacks against the protocols that satisfy that property. (Note that these attacks were described previously by various authors)

8.4 Incentive Driven Deviation

A miner is said to be honest iff he does the following at every time step t:

- Find A maximizing $S(A)$ among all blocks that the miner is aware of.
- For all owned coins c, attempt to mine a new block $B = M_P(A, c, t)$. If $B \neq \perp$ announce the new block B, otherwise do nothing.

Any profitable deviation from this will be considered an attack.

8.4.1 Predictable Selfish Mining

The attacker can withhold a newly mined block and start mining on top of it, if he can create a longer chain in private and release it later then the longest chain will contain more blocks from the attacker. In case of **predictability**, it can be efficiently computed whether or not the attacker will be able to produce the longest chain. In case of Local Predictability, one can compare his chain with the average rate of growth but in case of Global Predictability the attacker can exactly compute whether the attack will be successful or not.

Mitigation Snow White [1] and Ouroboros [7] provide proofs that any deviation from their prescribed protocol can only provide a small ε in additional mining rewards. However, [1] notes that it would be preferable for known attacks to be *strictly* disincentivized.

8.4.2 Predictable Double Spending

In a similar manner, the attacker can include a conflicting transaction in his secret longer chain which is revealed later.

Mitigation One defense against double spending attack could be to have a longer confirmation time (possibly beyond D)

8.4.3 Undetectable Nothing at Stake Attack

In case of an eventual fork, a miner could potentially mine on both the chains of the fork to double the chances of winning, this is known as a Nothing at Stake attack. In case of protocols having recency, it is not possible to detect this attack. We cannot "punish" suspected deviant miners without the risk of punishing honest but poorly-connected miners.

Mitigation Algorand [5] proposes a different approach: instead of using a longest-chain variant, it uses a Byzantine consensus protocol. Under some network connectivity assumptions, they show that the probability of a fork is negligible. As such, any deviant behavior that results in a fork (such as Undetectable Nothing-at-Stake) can be readily recognized as malicious, and safely ignored. Ethereum's Casper [2] proposes a third solution that they call "dunkles": punish every miner whose block winds up being orphaned (not a predecessor of the block maximizing $S(A)$). The high-level goal of this is to essentially copy the incentives from Proof-of-Work: if your block is orphaned, you still lose the electricity that went into mining it. By punishing the miner of every orphaned block, some honest miners will get punished just by bad luck, but it will also discourage attackers from mining off the longest chain. This seems like a promising direction, but there is currently no formal specification or rigorous evaluation of the proposal.

8.5 Novel Ideas

The paper presents a high level result that presents the challenge to design secure PoS protocols based on previously known attacks. They show that there are two categories of attacks which are complementary to each other, i.e., fixing one attack leads to vulnerability to attacks from the other category. The challenge then becomes to solve both categories of attacks together.

Acknowledgements

We would like to thank Shoeb Siddiqui for useful discussions on the above papers.

9

Brief Overview

Paper	Appeared in	Problem Statement	Solution Proposed
OHIE: Blockchain Scaling Made Simple	IEEE S&P 2020	Security-Scalability Tradeoff	Composes multiple parallel instance of Bitcoin's original backbone protocol
SPECTRE: A Fast and Scalable Cryptocurrency Protocol	Cryptology ePrint Archive	Security-Scalability Tradeoff	Utilizes a voting mechanism to resolve conflict sets.
PHANTOM and GHOSTDAG	Cryptology ePrint Archive	Security-Scalability Tradeoff	Emphasizes on the anticone for honest blocks to be limited in size to distinguish between blocks created by honest nodes and adversary.
The Bitcoin Backbone Protocol: Analysis and Applications	EUROCRYPT 2015	Validity of Nakomoto Consensus for solving the Byzantine Agreement Problem and obtaining bounds on performance of Bitcoin while maintaining validity in presence of adversaries.	It considers a bounded-delay model for the Bitcoin Network to prove security properties.
FruitChains: A Fair Blockchain	ACM Symposium on Principles of Distributed Computing 2017	Selfish Mining and centralization due to Mining Pools in Bitcoin	Proposes a fair blockchain protocol which is resistant to the selfish mining attack and reduces the incentive to form mining pools.

Table 2: **Protocol-based Papers**

Paper	Appeared in	Problem Statement	Solution Proposed
Nonlinear Blockchain Scalability: a Game-Theoretic Perspective	arXiv preprint	Blockchain Scalability	Presents an impossibility result that characterizes the tradeoff between (i) scalability; (ii) low finality-duration; and (iii) full verification.
Formal Barriers to Longest-Chain Proof-of-Stake Protocols	20th ACM Conference on Economics and Computation (2019)	Security Concerns in Proof-of-Stake Protocols	Presents a model and describes two complementary vulnerabilities in Proof-of-Stake Protocol

Table 3: **Protocol-based Papers**

List of Papers Summarized

[Read1] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

[Read2] Lin Chen, Lei Xu, Zhimin Gao, Keshav Kasichainula, and Weidong Shi. Nonlinear blockchain scalability: a game-theoretic perspective. *arXiv preprint arXiv:2001.08231*, 2020.

[Read3] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)*, pages 281–310. Springer, 2015.

[Read4] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324, 2017.

[Read5] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol, 2016. https://eprint.iacr.org/2016/1159.

[Read6] Yonatan Sompolinsky, Shai Wyborski, and Aviv Zohar. Phantom and ghostdag: A scalable generalization of nakamoto consensus. Cryptology ePrint Archive, Report 2018/104, 2018. https://eprint.iacr.org/2018/104.

[Read7] H. Yu, I. Nikolic, R. Hou, and P. Saxena. Ohie: Blockchain scaling made simple. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 112–127, Los Alamitos, CA, USA, May 2020. IEEE Computer Society.

References

[1] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *International Conference on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.

[2] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[3] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[4] Michael R Garey and David S Johnson. Computers and intractability, vol. 29, 2002.

[5] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017.

[6] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. Cryptology ePrint Archive, Report 2016/545, 2016. https://eprint.iacr.org/2016/545.

[7] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[8] Will Martino, Monica Quaintance, and Stuart Popejoy. Chainweb: A proof-of-work parallel-chain architecture for massive throughput, 2018.

[9] Serguei Popov, Olivia Saa, and Paulo Finardi. Equilibria in the tangle. *Computers & Industrial Engineering*, 136:160–172, 2019.

[10] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[11] Aviv Yaish. Phantom. https://github.com/AvivYaish/PHANTOM, 2019.