On the (De)centralization of FruitChains

Aikaterini-Panagiota Stouka, Nethermind
Thomas Zacharias, University of Glasgow

Bitcoin and PoW blockchain protocols

- Bitcoin and in general blockchain utilizes a distributed ledger of transactions that is extended and maintained in a decentralized manner without the need of a trusted party.
- Transactions are stored in blocks maintained by nodes in a peer–to-peer network.
- Blocks form a chain; every block has a reference to the previous block.
- If more than one candidate chains have been created, the longest chain will constitute the ledger.
- A new block is added to the ledger when a node, called miner, solves a Proof of Work Puzzle that demands computational power.

Bitcoin and PoW blockchain protocols

- The miner creates an input that includes among others the hash of the previous block and a fingerprint (Merkle root) of the transactions that need to be added to the ledger.
- After that it appends a nonce to the input and it computes its hash. If this hash is smaller than a predetermined value, then a new block has been produced. Otherwise, it repeats the same procedure using a different nonce.
- When a miner produces a block, new coins are minted and are given to the miner as compensation for its work.

Criticism on Bitcoin and the FruitChain proposal

- Bitcoin is vulnerable to selfish mining attacks:
  - The fraction of the blocks in the ledger that belong to the honest miners is reduced.
  - The attacker manages to replace the blocks of the honest miners with their own blocks.
  - An adversary with network advantage and 1/3 of total hashing power can launch a successful mining attack.
- Bitcoin has tendency to centralization:
  - Miners are organized into mining pools.
    - they solve computational puzzles of lower difficulty (partial PoW).
    - they get paid regularly according to the pool rules.
    - The leader of the pool may determine which transactions will be included in the next block.
  - In a pool, the miners' rewards have lower variance compared to solo mining.
  - Currently, only three pools constitute the majority of the computational power.

Criticism on Bitcoin and the FruitChain proposal

- Pass and Shi [PODC 2017] proposed **FruitChain**
  - Utilizes 2-for-1 PoW (introduced in the Bitcoin Backbone Protocol [EUROCRYPT 2015]).
  - Provably satisfies fairness (honest miners hold a fraction of blocks that is very close to their relative computational power).
  - Mitigates selfish mining attacks.
  - Reduces the variance as pools do (the miners get paid for solving puzzles of lower difficulty) but in a "fully decentralized way".

Common perception

The problem of centralization in PoW blockchain systems can be solved via lower rewards' variance, so in FruitChain the formation of pools is unnecessary

Common perception

Distribution of block rewards that is equitable makes the formation of mining pools redundant

Mining pools are unnecessary, because miners can produce fruits in short time

The problem of centralization in PoW blockchain systems can be solved via lower rewards' variance, so in FruitChain the formation of pools is unnecessary

There is no mean of pool existence in Fruitchain

High variance is the main motivation for joining a pool

When the parties' rewards are concentrated with high probability to their initial resources, the parties lose their motivation to form mining pools

Our contributions

Contrary to the common perception, we prove that lower variance of the rewards does not eliminate the tendency of the PoW blockchain protocols to centralization, as miners have also the incentive to create large pools for sharing the cost of creating the instance they need to solve the PoW puzzle.

Our contributions

We prove that there is a completely centralized equilibrium where all the miners form a unique pool whose pool leader is responsible for determining the instance that will be used for the PoW procedure.

The notion of equilibrium that we use is Equilibrium with Virtual Payoff (EVP)[AFT'21].

Our contributions

In order to be able to describe formally this equilibrium we provide also a formal definition of a pool in a blockchain system, as a subset of parties along with their communication setting and their execution guidelines.

We abstract the procedures of FruitChain as oracles and assign to each of them a cost.

Outline of FruitChain

- Miners store transactions in data structures called fruits (instead of blocks).
- Fruit mining has lower difficulty than block mining.
- 2-for-1 PoW: the miner computes hashes of a specific input, where the prefix and the suffix of the hash determine whether a block or a fruit has been mined, respectively.
- Fruits are stored in blocks and they need to be recent i.e., every fruit points to a block not far from the latest block.

The core arguments in FruitChain security

- Selfish mining attacks are prevented because even if an attacker withholds a block, the fruits of this block that are still recent can be stored in a later block.
- Due to fruit recency, an attacker is not able to precompute an excessive amount of fruits and reveal it later, thus disrupting the chain quality of the protocol.

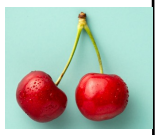
FruitChain parameters

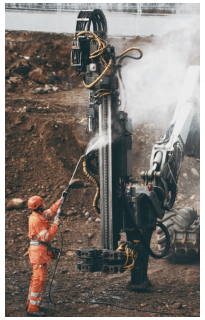
- A random oracle $H(\cdot)$ that responds to (block and fruit) mining queries.
- A collision resistant hash function (CRHF) $d(\cdot)$, utilized to digest sets of fruits.
- A block mining hardness parameter p_b.
- A fruit mining hardness parameter p_f.
- A recency parameter r that determines how far back can a fruit "hang".

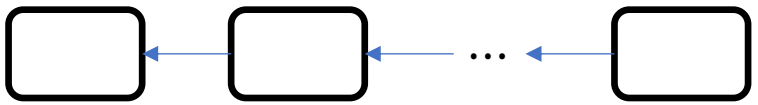
FRUITCHAIN in our framework

- The longest chain oracle O_{lc}
  - Updates the longest valid chain.
  - Outputs the new fruit and block pointers and a sequence of records.
- The fruit set oracle O_{fs} outputs  and $d($  $)$.
- A transaction oracle O_{tx} outputs the record of all valid transactions.
- A new instance consists of
  - The block pointer;
  - The fruit pointer;
  - A random nonce;
  - d () ;
  - The record of valid transactions.
- A random oracle O_{ro} responds to new instance queries.

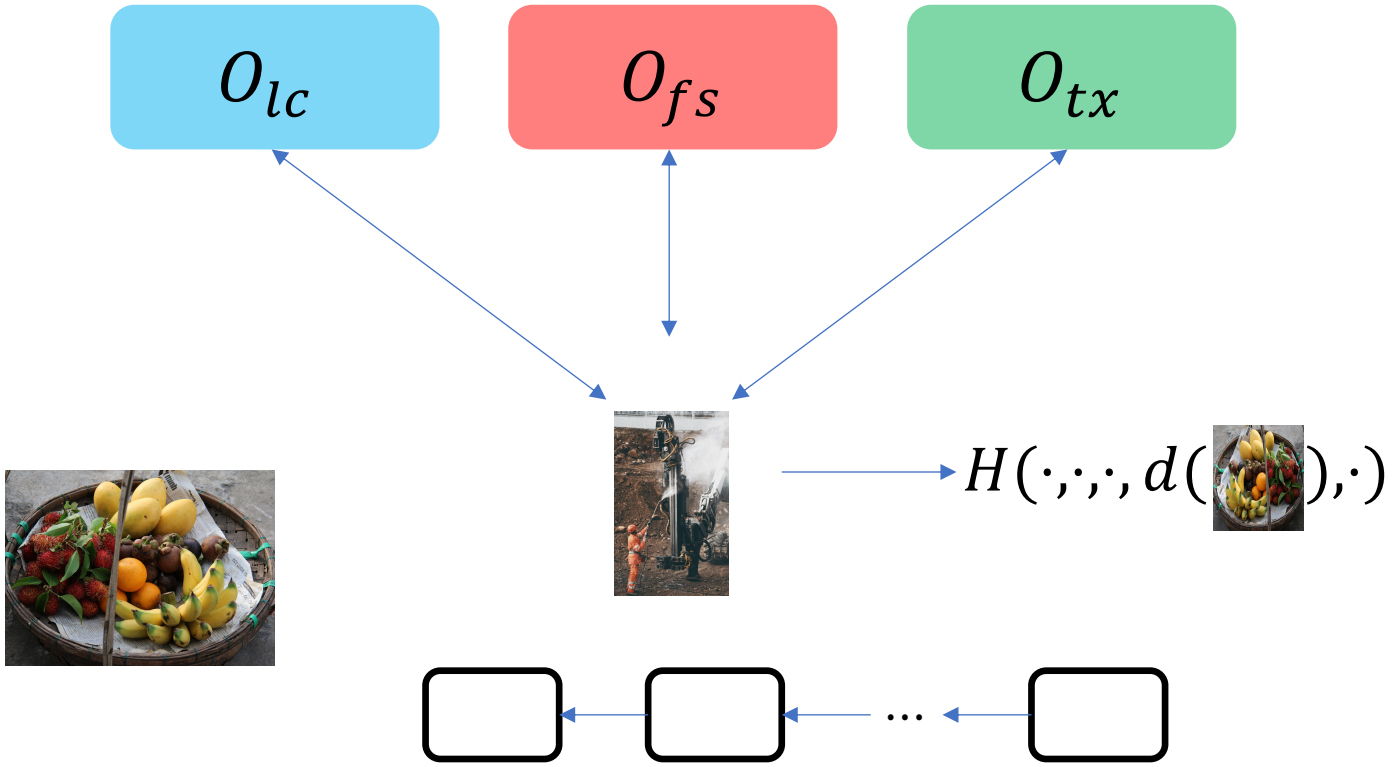
Diffuse





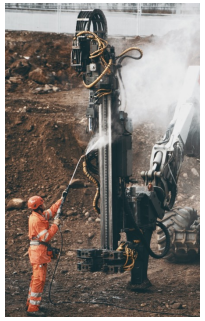


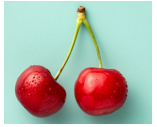
Diffuse

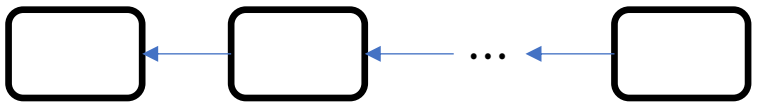


Diffuse

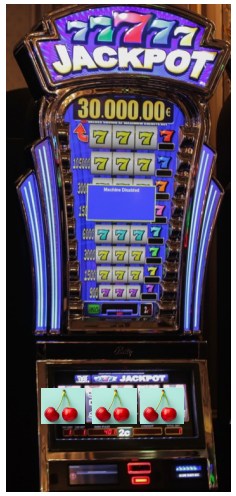
$H(\cdot,\cdot,\cdot, d(\ \ ),\cdot)$

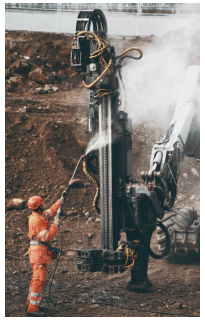


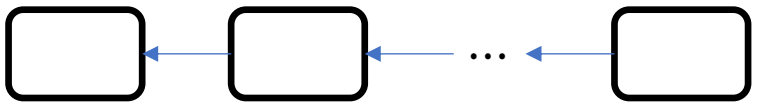




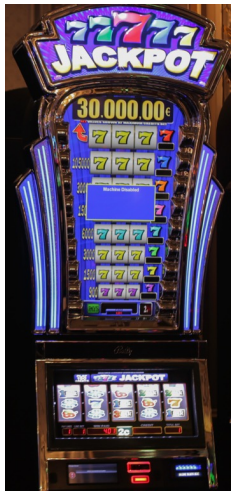
Diffuse



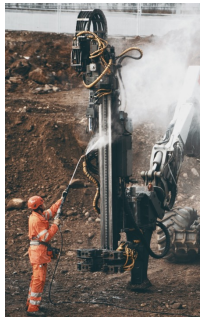


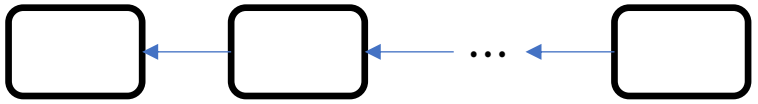


Diffuse

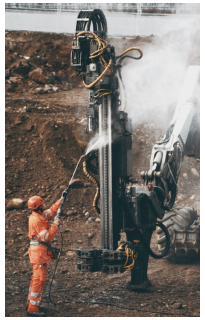


$H(\cdot,\cdot,\cdot,d($  $),\cdot)$



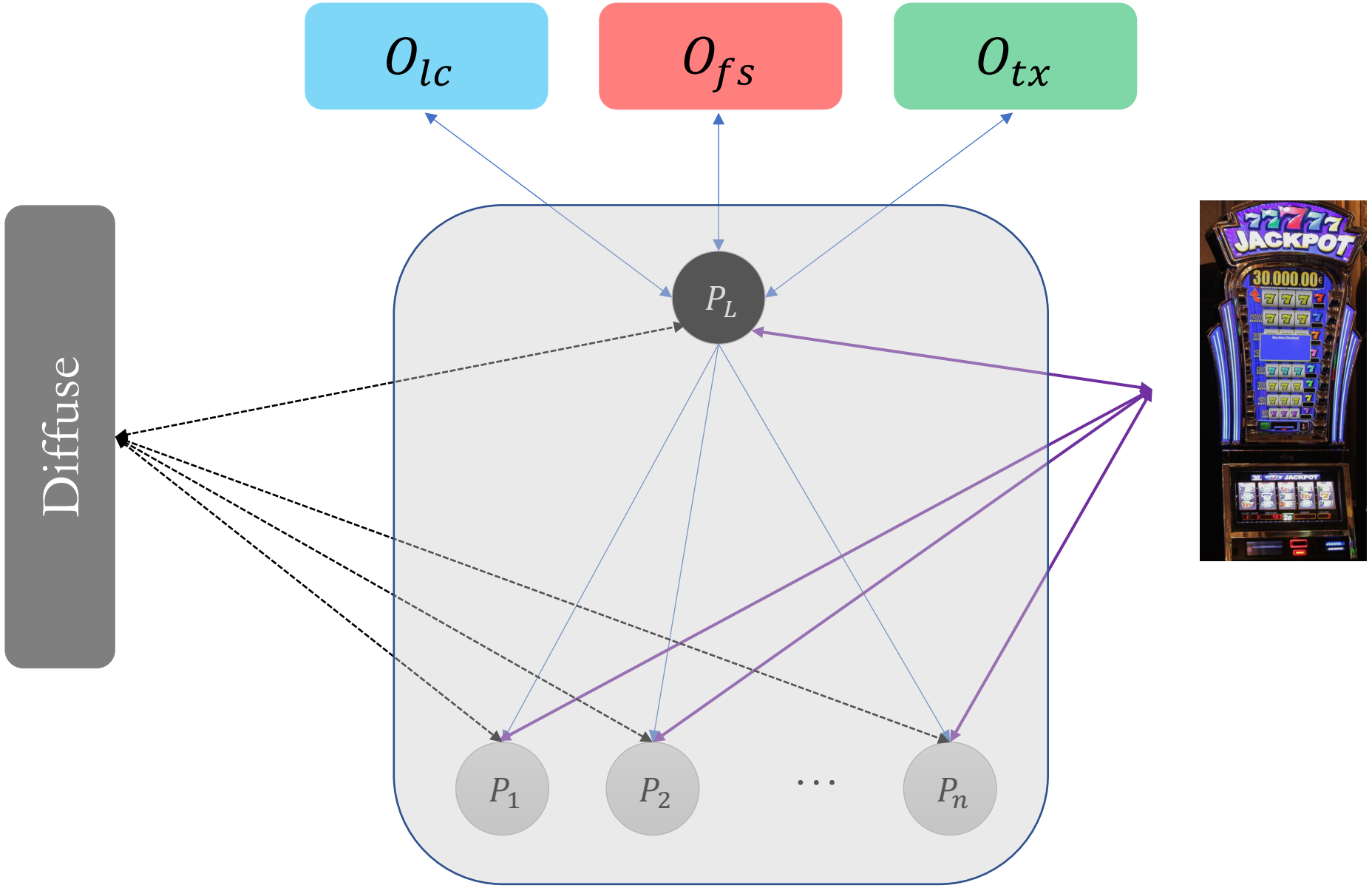


Diffuse



A centralized pool for FruitChain

- During each round, the pool leader asks the longest chain oracle, fruit set oracle, and transaction oracle, and creates the instance components (apart from the nonce) that will be used for the queries to the random oracle.
- Then, it sends this instance to the pool members.
- The pool leader and the other members ask the random oracle q queries when they are activated, each time refreshing the nonce.
- When a fruit or a block is produced, it is sent to the Diffuse functionality (at most one block per round).


O_{lc}
O_{fs}
O_{tx}
Diffuse
P_L
P_1
P_2
...
P_n
JACKPOT
30.000,00

Payment rules and compliance checks

- If the cost that the pool leader incurred for creating the instances is higher than the block's rewards, then the pool leader holds all the rewards.
- If the block's rewards are higher, then the pool leader subtracts the cost and shares the remaining rewards equally among all the members of the pool including itself.
- The pool leader and other pool members check if the diffused fruits and blocks use the instance sent by the pool leader (else, they abandon the pool).
- The pool members check if the payments have been computed correctly (else, they abandon the pool). Formally, this check is abstracted via a query to a light transaction verification oracle O_{ltx} that adds a cost C_{ltx} per round.

Some of the deviations that we examine in our proof

- A coalition of members ignore the PoW instance provided by the pool leader and compute their own instance (extend another chain and/or include different transactions or fruits)
- A coalition of members (1) abandon the pool and (2) remain idle or set up their own pool with different payment and compliance rules or mine on their own.
- A coalition of members ask fewer queries the relevant oracles.
- The leader dissolves the pool and mines on their own.
- The leader does not pay the members correctly by creating incorrect instance.
- The leader collaborates with some other members in order to not pay some other members.
- The leader with/or some members do not send to the Diffuse function some of their fruits or their block.

Equilibria with Virtual Payoff (EVPs) - Kiayias and Stouka [AFT 2021]

- An environment $\mathcal{Z}$ schedules the protocol execution and provides parties with their inputs.
- An adversary $\mathcal{A}$ controls a coalition of up to t parties.
- The rewards are virtual: each honest party can have a different view on the utility of each other party and thus, the utility of the adversary.
- We compare the following utilities among all parties' local views:
  - The lowest utility $U_{\mathbf{C}}^{\min}(\mathcal{E}_{\mathcal{Z},\mathcal{H}_{\mathbf{C}}})$ when the coalition honestly follows the protocol.
  - The highest utility $U_{\mathbf{C}}^{\max}(\mathcal{E}_{\mathcal{Z},\mathcal{A}})$ where the adversary behaves arbitrarily.

Equilibria with Virtual Payoff (EVPs) – Kiayias and Stouka [AFT 2021]

Let ϵ, ϵ' be non-negative real values and κ be the security parameter. A protocol Π is a (t, ϵ, ϵ')-EVP if for every PPT adversary that controls a set $\mathbf{C}$ of at most t parties, it holds that

$$U_{\mathbf{C}}^{\max}(\mathcal{E}_{\mathcal{Z},\mathcal{A}}) \leq U_{\mathbf{C}}^{\min}(\mathcal{E}_{\mathcal{Z},\mathcal{H}_{\mathbf{C}}}) + \epsilon \cdot \left| U_{\mathbf{C}}^{\min}(\mathcal{E}_{\mathcal{Z},\mathcal{H}_{\mathbf{C}}}) \right| + \epsilon'$$

with 1-negl(κ) probability.

The centralized pool as an EVP

Let N be the number of rounds and n be the number of parties. Let R_f be the rewards per fruit, C_{lc} be the cost of a query to the longest chain oracle, C_{ltx} be the cost of a query to the light transaction verification oracle, and q be the number of RO queries (per party per round).

Assume that the block mining hardness parameter is $p_b = \Omega(\frac{1}{nq})$ and the fruit mining hardness parameter is $p_f < \frac{1}{2}$. Then, the FruitChain centralized pool is a $(n-1, 0, \epsilon')$-EVP, where ϵ' is dominated by the following three additive approximation factors:

1. $O\left((N \cdot n)^{\frac{3}{4}} \cdot \log\kappa \cdot p_f \cdot R_f\right)$
2. $O\left(N^{\frac{1}{2}} \cdot \log\kappa \cdot C_{lc}\right)$
3. $O(N \cdot n \cdot \log\kappa \cdot C_{ltx})$

The centralized pool as an EVP

- Dominant additive approximation factors
  - $O\left((N \cdot n)^{\frac{3}{4}} \cdot \log\kappa \cdot p_f \cdot R_f\right)$: appears because the EVP notion compares the exact profit of the honest and adversarial strategy with overwhelming probability.
  - $O\left(N^{\frac{1}{2}} \cdot \log\kappa \cdot C_{lc}\right)$: is due to the same reason as above.
  - $O(N \cdot n \cdot \log\kappa \cdot C_{ltx})$: the honest parties check the validity of the payments (typically, this cost is small).

Discussion and key takeaways

- Our results indicate that, apart from reducing the variance of the rewards, further research is needed to incentivize decentralization in PoW protocols.
- Other proposals (e.g., Smartpool [USENIX 2017], StrongChain [USENIX 2019]) where miners validate the transactions, have similar tendency to centralization (i.e., collusion to share verification costs).
- Candidate mitigation strategy: construct PoW puzzles that disincentivize the formation of pools, while being applicable to blockchain protocols that satisfy fairness (like FruitChain)
  - Promising starting point: GSCS [Wang et al., IEEE Access 2020] deploys a non-parallelizable PoW puzzle used in a consensus mechanism that guarantees fairness.

On the (De)centralization of FruitChains

Aikaterini-Panagiota Stouka, Nethermind

Thomas Zacharias, University of Glasgow