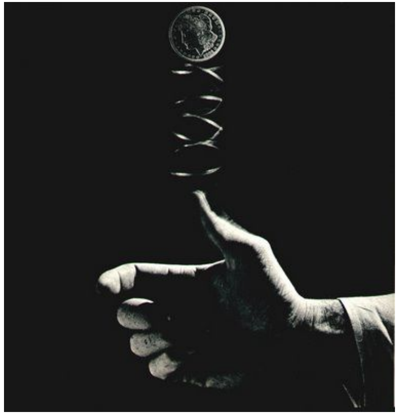
Formal Barriers to Proof-of-Stake Protocols

Jonah Brown-Cohen, Arvind Narayanan,
Christos-Alexandros Psomas, S. Matthew Weinberg

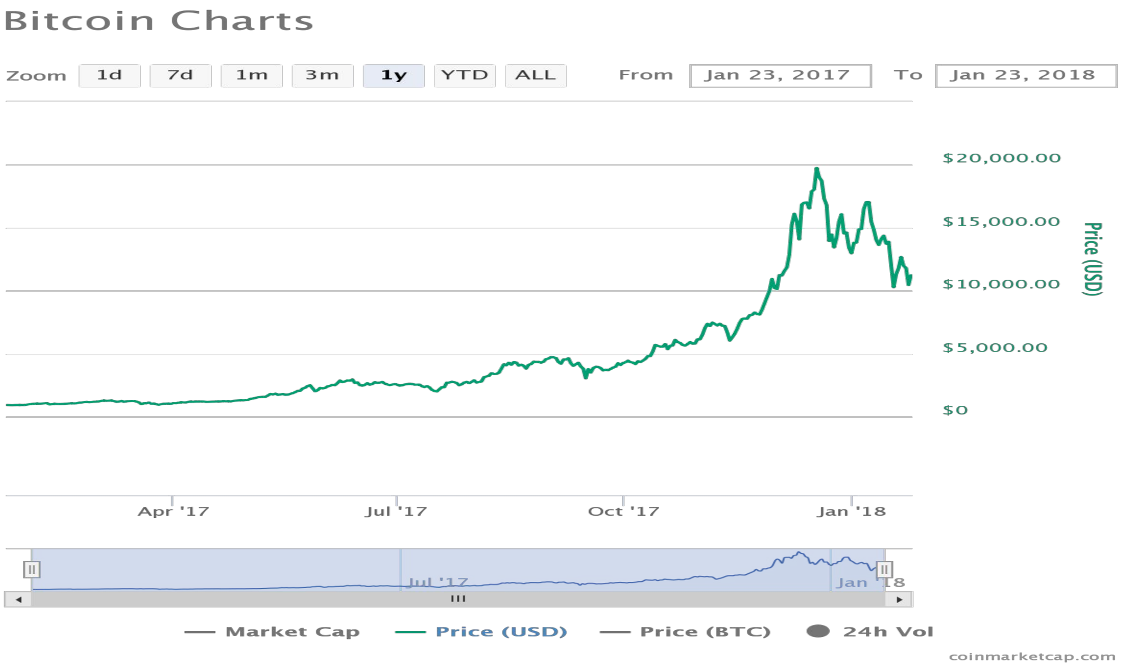
Proof-of-Stake

- Random miner selected with probability proportional to wealth rather than computational power
- “One coin, one vote” rather than “one cpu, one vote”
- Question: can similar security be achieved in this setting?



Incentive-driven Analysis

- All participants act strategically to maximize revenue



Talk Overview

1. A model for analyzing incentives in Proof-of-Stake protocols
2. A set of properties, such that every protocol in the model satisfies at least one of the properties
3. For each property, incentive-driven attacks against protocols satisfying that property

Disclaimer

The attacks we describe are well-understood, the fact that they apply to a broad class of protocols is our main contribution

Intuition for Results

- Need a source of pseudorandomness to pick random miner
- Proof-of-Work: randomness comes from brute force guessing of nonce
- Proof-of-Stake: randomness comes from the protocol itself
- Miners are incentivized to attempt to influence the randomness by deviating from the protocol

Talk Overview

1. **A model for analyzing incentives in Proof-of-Stake protocols**
2. A set of properties, such that every protocol in the model satisfies at least one of the properties
3. For each property, incentive-driven attacks against protocols satisfying that property

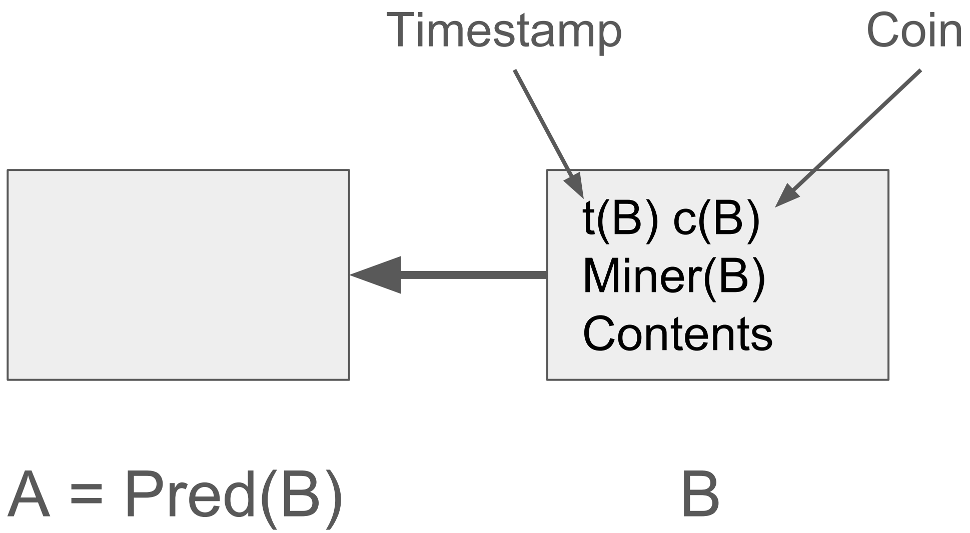
The Model

High Level Idea

1. Protocol uses some method to pick a coin
2. Protocol uses some method to pick an existing block
3. Owner of the coin gets to add a new valid block of transactions on top of the existing block
4. Repeat

The Model

Blocks



The Model

Coins

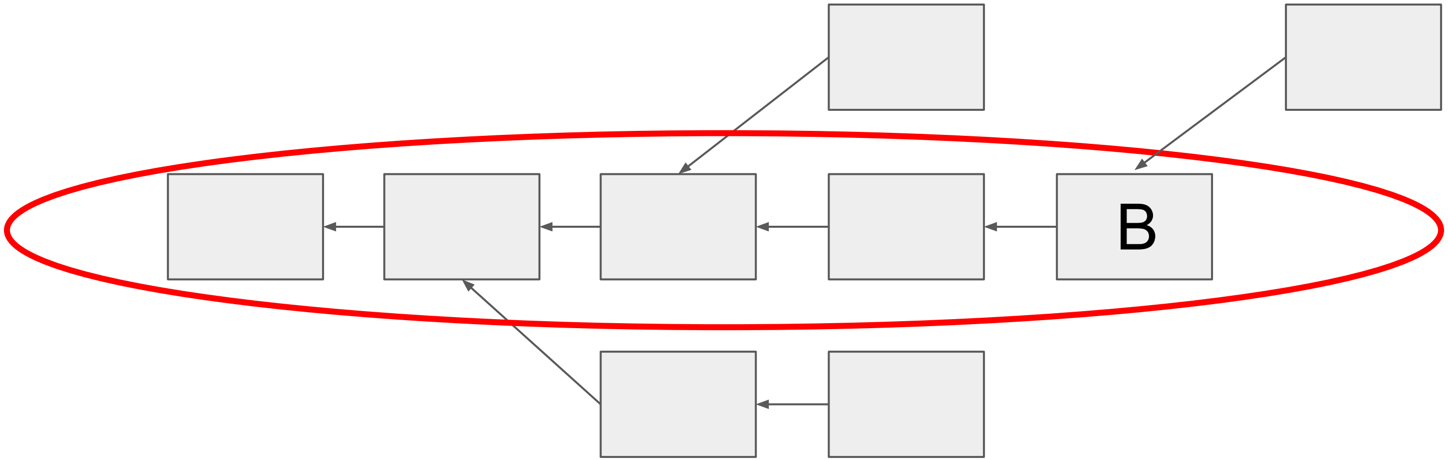
- For a block B, history of transactions in B and B’s predecessors define ownership of coins by protocol participants
- Owner(c) at B denotes participant who owns c in history defined by B and B’s predecessors

The Model

Assumptions

1. Chain Dependence: Validity of block B at time t depends only on t and the predecessors of B
2. Monotonicity: If B is valid at time t then it is valid at all future times t’ > t

The Model



The Model

Justifying Assumptions

- Assumptions hold for Bitcoin and other Proof-of-Work protocols
- Eclipse attacks: without assumptions an attacker can withhold messages to convince a victim invalid blocks are in fact valid.

The Model

Protocol

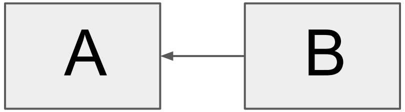
A Proof-of-Stake protocol is defined by two functions

1. A *validating function* V which takes as input a block and outputs 0 or 1
2. A *mining function* M which takes as input a block B, a coin c and a timestamp t, and outputs a block

The Model

Validating Function

1. V must be efficiently computable by every protocol participant
2. A block B with Pred(B) = A is valid at time t if and only if
   a. V(B) = 1
   b. Miner(B) = Owner(c(B)) at A
   c. $t(A) \leq t(B) \leq t$



The Model

Mining Function

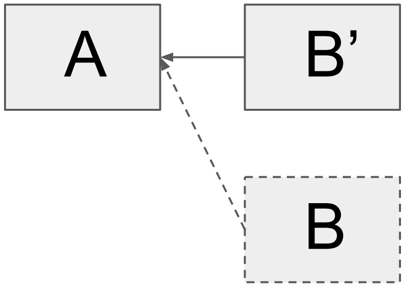
1. M(A, c, t) is efficiently computable by Owner(c) at A
2. If there is a block B valid at time t with c(B) = c and t(B) = t, and Pred(B) = A, then:

M(A, c, t) = B’, where B’ satisfies all the above properties of B

3. If there is no B as above, then M(A, c, t) = $\perp$

The Model

$$M(A,c,t) = B'$$



The Model

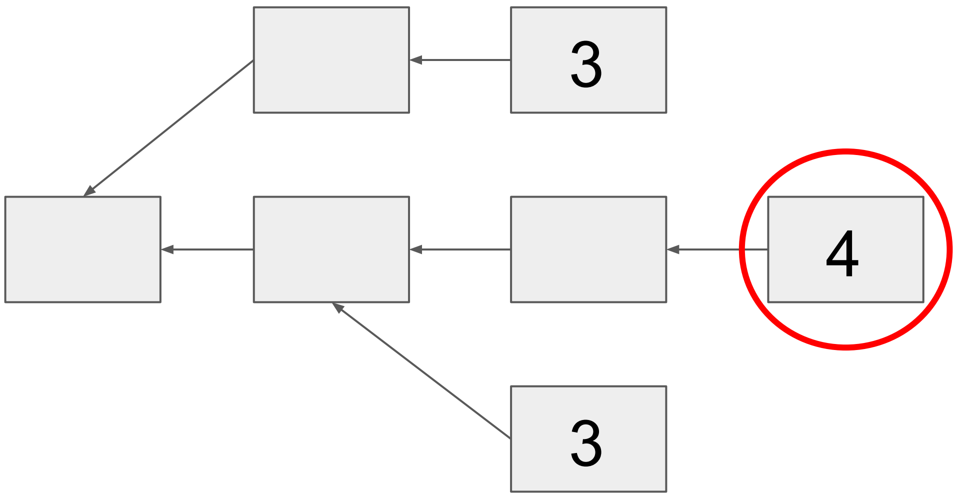
Longest-Chain Protocol

A Longest-Chain protocol has a scoring functions S which takes as input a block and outputs a monotone increasing score:

If A is the predecessor of B then $S(A) < S(B)$

Miners are supposed to mine on top of A maximizing $S(A)$

The Model



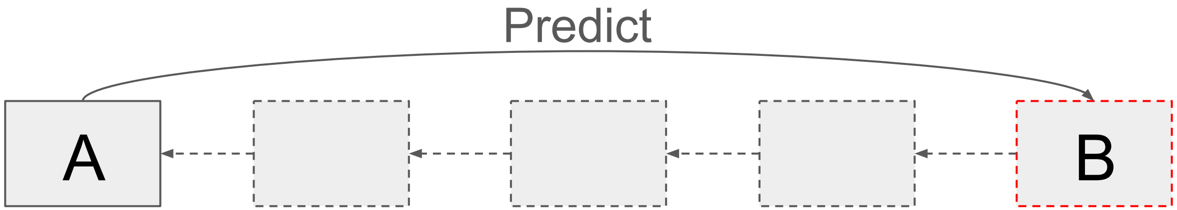
Talk Overview

1. A model for analyzing incentives in Proof-of-Stake protocols
2. **A set of properties, such that every protocol in the model satisfies at least one of the properties**
3. For each property, incentive-driven attacks against protocols satisfying that property

Properties of Protocols

D-Locally Predictable

For a coin c, Owner(c) can efficiently predict D blocks in advance if she is eligible to use c to mine a block



Properties of Protocols

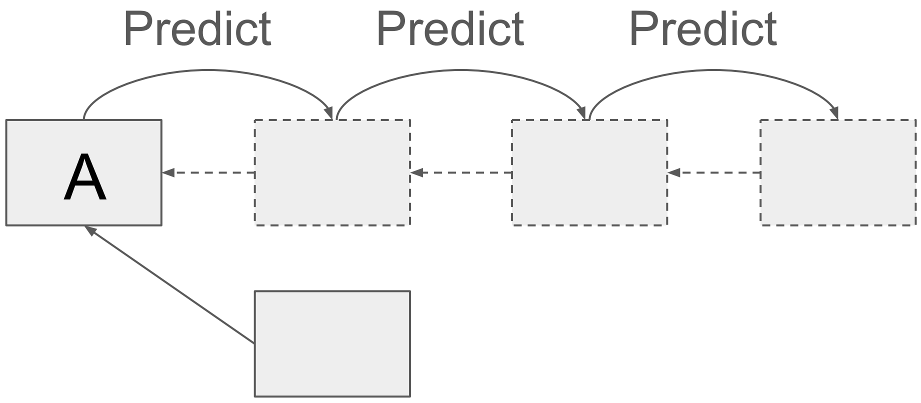
Observation

Every Proof-of-Stake protocol is 1-locally predictable

Proof. Just use the mining function M to efficiently predict whether you can mine the next block.

Properties of Protocols

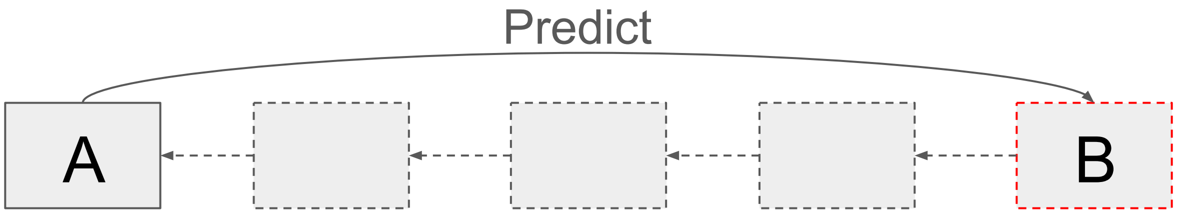
1-local predictability can be chained on a private fork



Properties of Protocols

D-Globally Predictable

Every protocol participant can efficiently predict D blocks in advance whether Owner(c) is eligible to use c to mine a block.



Examples

Let T be some threshold and H be a hash function

- A protocol where $V(B) = 1 \Leftrightarrow H(c(B),t(B)) < T$
- A protocol where
  - Every block contains a signature $s(B)$
  - $M(A, c, t)$ outputs B with $s(B) = SIG(H(s(A), t))$ where $SIG()$ is the digital signature of $Owner(c)$
  - $V(B) = 1 \Leftrightarrow H(s(B)) < T$

Properties of Protocols

D-Recent

The negation of D-locally predictable. Miner(c) **cannot** efficiently predict D blocks in advance if she is eligible to use c to mine a block

Eligibility to mine a block depends on “recent history”

Talk Overview

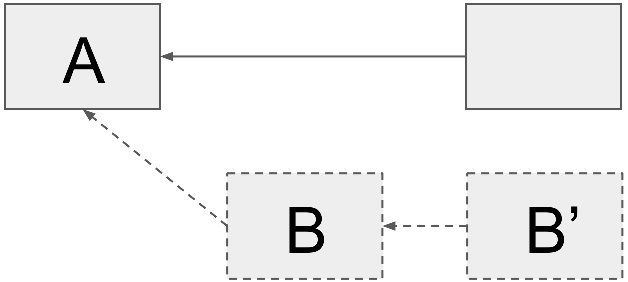
1. A model for analyzing incentives in Proof-of-Stake protocols
2. A set of properties, such that every protocol in the model satisfies at least one of the properties
3. **For each property, incentive-driven attacks against protocols satisfying that property**

Attacks

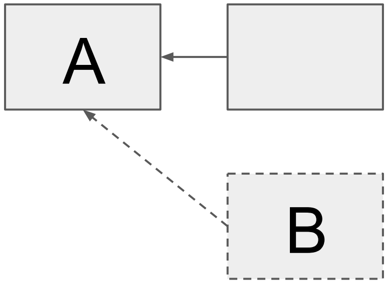
Predictable Selfish Mining

- Original selfish mining attack: withhold a newly mined block B and secretly try to mine on top of it
- If you mine another block B’, then you have the longest chain, even if other miners mine a block on Pred(B)
- Risk: if other miners mine on top of Pred(B) first, your withheld block B may not be included in the longest chain

Attacks



Attack Succeeds



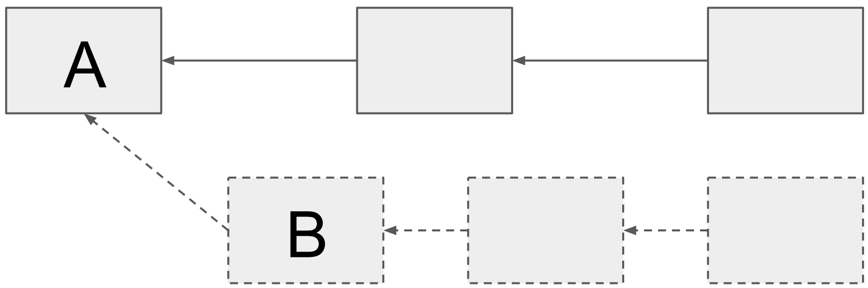
Attack Fails

Attacks

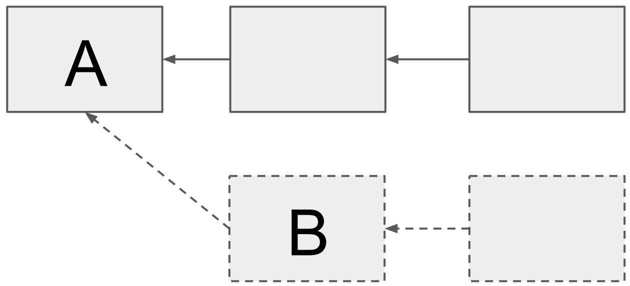
Predictable Selfish Mining

- With global predictability there is no risk!
- Can predict precisely when you are able to mine k blocks faster than the rest of the miners

Attacks



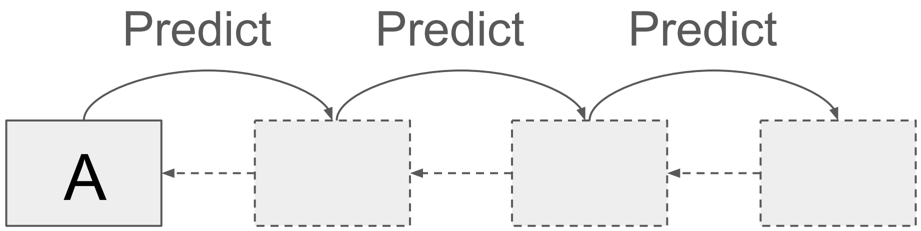
Launch Attack



Abort Attack

Predictable Selfish Mining

- Even with 1-Local Predictability there is reduced risk
- Can predict precisely how fast you will mine k blocks and then compare to the average rate

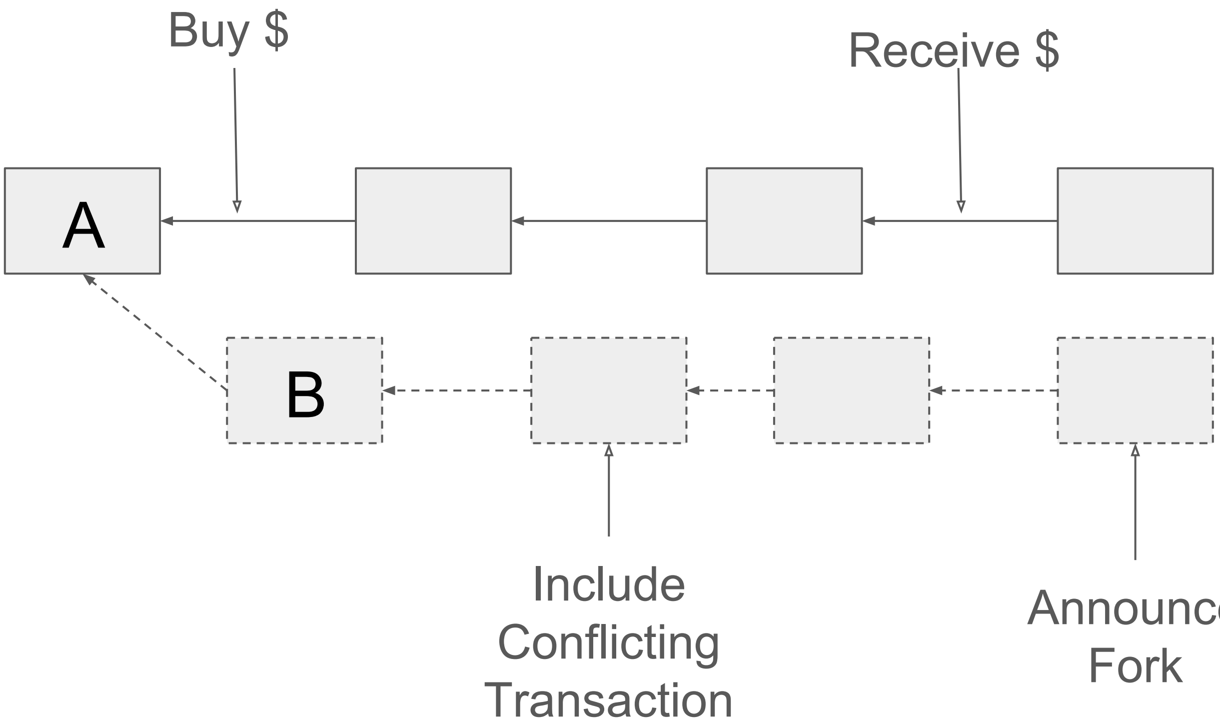


Attacks

Predictable Double Spending

- If you predict that you can produce a secret fork of k blocks faster than the rest of the miners
- Buy dollars using coins from the protocol
- Include a conflicting transaction in your secret fork
- If dollars are delivered before k blocks are mined, reveal your k secret blocks to cancel the transaction

Attacks


Buy $
Receive $
A
B
Include Conflicting Transaction
Announce Fork

Attacks

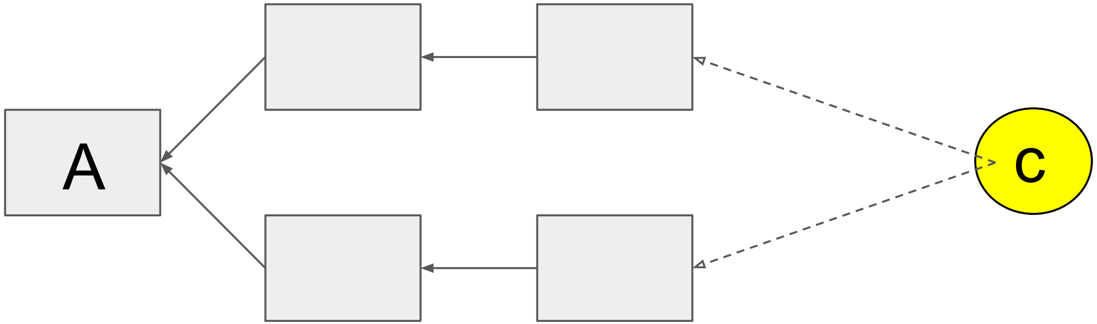
- D-Predictability for large values of D allows you time to prepare for launching an attack
- Could be useful to prepare for double-spending
- Could allow you to prepare for other stranger attacks e.g. by offering to take a bribe to create a fork

Attacks

Undetectable Nothing-at-Stake

- For D-Recent protocols, blocks A and B at the two ends of a length D fork have “independent pseudorandomness”
- Attempting to mine on both sides of the fork doubles your chances of successfully mining
- If all your coins are held by separate accounts, can make this attack undetectable

Attacks



Existing Protocols

- D-Globally Predictable
  - Cardano/Ouroboros (for large D)
  - Peercoin (for all D)
  - Tezos (for large D)
- 1-Locally Predictable and 2-Recent
  - Algorand (not longest chain)

Conclusion

- There are incentive-driven security issues for Proof-of-Stake protocols not present in Proof-of-Work
- 1-Local Predictability is necessary
- There is a tradeoff between predictability and recency
- Global Predictability is NOT necessary

Thanks!