Bribe-resilient NIPoPoWs

Changing Blockchain Macroeconomic Policy through Soft Forks

Kostis Karantias, Dionysis Zindros

ATHECRYPT 2021
Superblocks

- Superblocks allow the construction of superlight clients
- Superlight clients are exponentially more efficient light clients
- They are built on top of NIPoPoWs = Non-Interactive Proofs of Proof-of-Work
\( H(B) \leq T \)
\[ \text{Pr}[H(B) < T/2^\mu \mid H(B) < T] = 1 / 2^\mu \]
level $\mu = 0$

level 1

level 2

level 3
Proofs

- Verifier wants to deduce most recent $k = 6$ blocks of honest chain
- Pick highest $\mu$ with at least $m = 128$ $\mu$-superblocks
- Proof with most $\mu$-superblocks wins

Does $\pi^H$ have more $\mu$-superblocks than $\pi^A$?
just send these three blocks as representatives
Bribing attacks against superblocks -- withholding

- NIPoPoW protocols are secure in the cryptographic/backbone model
- What about incentives? Rational model
- NIPoPoWs can allow the transfer of large amounts of money
- A simple bribing attack can break the protocol:
  - Adversary pays rational miners to *withhold* superblocks
  - If block reward is R, adversary pays rational miner R+ε to *withhold* superblock
  - This *biases* the chain of honest parties to have fewer superblocks per blocks
  - The same attack applies in simple Bitcoin, but is *much* more expensive
- Incentive attack discovered by Benedict Bünz et al. from Stanford
Reason for bribing attacks

- Bitcoin is also susceptible to bribing attacks
- If block reward $\times k \ll$ value transferred
- But superblocks get same reward as regular blocks
- Withholding $m$ superblocks allow adversary to hide $\mu$-superchain at any level
- Superblock bribing attack stems from all blocks having the same reward
- Can we change the reward of blocks?
Idea! Superblocks should pay out more

- If simple block reward is $R$, then $\mu$-superblock should be rewarded with $2^\mu R$
- That way, superblock bribes cost the same as regular bitcoin bribing
Review: Hard and soft forks

Soft fork

- **Reduces** the validity language
- txs / blocks that were valid are now invalid
- All old invalid txs / blocks are still invalid
- Old miners **accept** new-style txs / blocks
- New miners **reject** some old-style txs / blocks
How to soft fork?

- Seems difficult to make blocks pay out *more* with a soft fork…
- After all, why would unupgraded miners accept blocks paying out more as valid?
- Approach: Pay out exactly the same as before
- Use a *smart contract beneficiary* to receive these payouts
- Miners themselves get paid only later
- **Require** valid blocks to pay out to **this** smart contract only

* this idea was pioneered by SmartPool
Conditions for paying out

- Smart contract must have enough money to perform payout
- Lower reward of regular block so that superblocks can be paid out

Example schedule:

- Regular block reward $R'$ becomes $1/10$ of old block reward $R$
- $\mu$-superblocks receive $2^\mu R'$

With a few minor adjustments:

- $\mu$-superblocks are only rewarded after the first $m$
- $R$ drop to a half every time a power of 2 blocks have passed
Conditions of applicability

Expectation of payment must be upper-bound by previous policy:

\[
\text{old average payment per block} \geq \text{new average payment per block}
\]

What happens if contract becomes bankrupt?
No problem, wait until it receives more rewards, pay debts later
contract SuperBlocks {
    uint256 constant public R = 0.5 ether;
    uint256 constant public m = 128;
    mapping(address => uint256) balances;
    mapping(uint256 => bool) blockClaimed;
    mapping(uint256 => uint256) muCount;

    function () external payable {
        // just accept the payment
    }

    function claimBlock(int blockNumber, int realMu, int claimedMu) {
        // must be called within a 256-block window
        uint256 blockHash = blockhash(blockNumber);
        // require miner to annotate block by including a first tx
        require(getBlock(blockNumber).transactions[0].sender == msg.sender);
        require(blockClaimed[blockNumber]);
        // check superblock level of block
        require(blockHash >> realMu == 0);
        require(blockHash >> (realMu + 1) > 0);
        require(claimedMu <= realMu);
        muCount[realMu]++;
        require(muCount[claimedMu] >= m);
        blockClaimed[blockNumber] = true;
        // allocate reward
        balances[msg.sender] += R * 2^claimedMu;
    }

    function withdraw(uint256 amount) {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        msg.sender.send(amount);
    }
}
Other applications

Blinded mining

- Reward miners for later revealing a hidden commitment in a block
- Useful building block for variable difficulty NIPoPoWs
- (Superblock level is hidden until block becomes confirmed)
Other applications

Smooth emission

- Block rewards half every 4 years on bitcoin
- There are other schedules for other coins
- This sudden change in rewards has chaotic influence on the economy
- Some coins emit “smoothly”
- i.e., change the amount of reward per block slowly
- We can change a sudden emission coin to a smooth emission coin with a soft fork
- The smart contract acts as a low-pass filter
Non-applications

Difficulty bombs

- Increase difficulty of blocks
- While it is possible to do with a soft fork in this manner, it is insecure
Other applications

Ensuring availability -- online miners

- Require miners to answer questions about data in block
- Only after, say, 1000 blocks of availability, reward miner
- If block data is withheld and proof of it can be presented, remove miner reward
Thank you! Questions?
References

- Bribe-resilient NIPoPoWs
  Kostis Karantias, Dionysis Zindros
  pending submission to ePrint, awaiting conference review

- Non-Interactive Proofs of Proof-of-Work
  Aggelos Kiayias, Andrew Miller, Dionysis Zindros
  FC 2020

- The Velvet Path to Superlight Blockchain Clients
  Aggelos Kiayias, Andrianna Polydouri, Dionysis Zindros
  ePrint pre-print 2020

- Compact Storage of Superblocks for NIPoPoW Applications
  Kostis Karantias, Aggelos Kiayias, Dionysis Zindros
  MARBLE 2020, nominated for Best Paper Award

- Variable-difficulty NIPoPoWs
  Aggelos Kiayias, Nikos Leonardos, Dionysis Zindros
  pending pre-print