ADESS: A Proof-of-Work Protocol to Deter Double-Spend Attacks

Daniel J. Aronoff[1] and Isaac Ardis[2]

[1] **Massachusetts Institute of Technology**

[2] **Ethereum Classic Labs**

February 2024

The Problem: PoW Blockchains are Vulnerable to Double-Spend Attacks

Nakamoto Bitcoin White Paper (2008) - claims double-spend attacks are deterred

- Cost of hashrate
- Risk of currency depreciation

Nakamoto's claims don't stand up (Gervais (2016), Budish (2018))

- Successful attacker receives block rewards
  $\hookrightarrow$ net cost can be small
- Attacker can profit by shorting the cryptocurrency

Moroz et.al. (2020): Victim can retaliate by building incumbent chain
$\hookrightarrow$ leads to War of Attrition. Raises cost of attack

However...

- There are equilibrium strategies that can support high and low cost attacks
- Victim may not have access to hashrate

✓ Existing PoW protocols provide limited deterrence to double-spend attacks

An Improvement: *ADESS* Increases the Cost of a Double-Spend Attack

What *ADESS* Is

2 modifications to the Nakamoto (PoW) protocol:

- **Identification** identifies attacker chain by comparing temporal sequence of blocks on competing chains
- **Penalty** discounts weight of blocks on attacker's chain

How *ADESS* Works

- **Assignment of Attacker Chain** The first fork chain observed to have α blocks is canonical
  $\hookrightarrow$ Relies on attacker hiding blocks and victim's tradeoff btw security and discounted value of transaction (formalized by Gao and Ren (2022)). Can be calibrated to ensure canonical chain is never assigned the penalty (may be no penalty assignment)
- **Penalty Function** Scoring switches from cumulative puzzle difficulty to block count

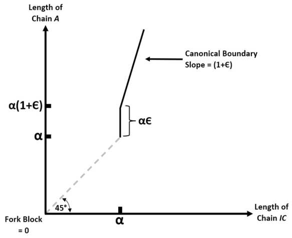
  (i) Penalty weight $\frac{1}{1+\xi}$ applies to attacker chain
  $\hookrightarrow$ attacker needs $(1+\xi)N$ blocks to overtake incumbent canonical chain at N post-fork blocks

  (ii) Scoring switched back to Nakamoto when attacker chain overtakes incumbent chain. Attacker cumulative puzzle difficulty score is set $=$ to incumbent $+\ \epsilon$ to give it slight lead

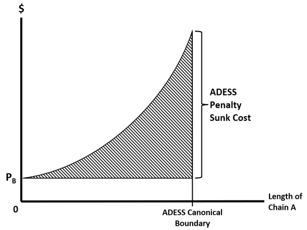
What *ADESS* Does

Key Insight The attacker has to build chain faster than incumbent canonical chain

Under penalty and complete puzzle adjustment after each block implies (ex-ante) a factor of $(1+\xi)^N$ units of hashrate to achieve score of N blocks on canonical chain



Penalty Function



EX-Ante Exponentially Increasing Hashrate Cost

$\hookrightarrow$ penalty function requires attacker to apply exponentially increasing hashrate

Note *ADESS* works best when penalty adjustments are frequent, e.g. Eth Classic

✓ *ADESS* increases the cost of a double-spend attack

Key Results of the Model

Proposition 1: The Increased Cost of Attack Under *ADESS*

The ex-ante expected hashrate required to carry out a double-spend attack under ADESS, where the penalty is bounded away from zero, i.e. $\xi > e$ for some $e > 0$, is weakly greater than the ex-ante expected hashrate required to carry out a double-spend attack under Nakamoto

Theorem 1: ADESS Bound on Double-Spend Profitability

For any transaction value v, there is a value of the penalty parameter $\underline{\xi}$ above which a double-spend attack is unprofitable

Corollary 1

For any penalty parameter $\xi > 0$ there corresponds an interval of transaction values $[0, v)$ for which a double-spend attack is unprofitable

✓ The baseline model assumes full puzzle adjustment after each block and no latency

Extensions of Baseline Model

- **Incomplete or infrequent puzzle adjustment**

  Incomplete implies penalty weight of $(1 + \beta\xi)$ for $\beta \in (0, 1)$

  Infrequent reduces the penalty, but does not eliminate it (except if attack is carried out inside of an epoch)

- **Network latency**

  Latency can cause a breakdown in consensus over the temporal order of blocks leading to disagreement over whether the attacker chain has crossed the canonical boundary. The attacker can maintain consensus by growing chain a exp rate past block propagation delay upper bound $\triangle$

- **Security of *ADESS* vs Nakamoto**

  ADESS attacker must apply exponential hashrate to make some nodes observe it as canonical. Thereafter, no cost

  Nakamoto attacker must indefinitely eclipse some nodes

  Tradeoff ADESS costs front-loaded. Nakamoto costs indefinite

- **Reliance on Temporal Observations**

  An *ADESS* node not connected to network when fork occurs cannot discern the canonical chain - a weakness compared to Nakamoto

✓ However, there are incentives for miners and others to maintain multiple nodes & verify state to entering nodes