Common Ancestor: A Composable Tie-Breaking Property and Its Application in Defending Against Stubborn Mining Attacks

Zibo Xu[1,a)] Akira Sakurai[1,b)] Taishi Nakai[1,c)] Kazuyuki Shudo[1,d)]

Abstract: In blockchain systems, resolving ties between competing forks is a critical part of maintaining consensus and security. However, we found that existing tie-breaking rules(e.g. Smallest-Hash, Last-Generated, PPoW) are vulnerable to attacks that exploit long-lived forks precisely because they typically decide at the chain tip: an adversary can repeatedly recreate ties at the tip and induce oscillations and unstable preferences. To address this limitation, we propose the Common Ancestor (CA), a tie-breaking property which resolves ties based on the block where a fork begins means the outcome can be reused for the lifetime of the fork, suppressing oscillations. CA also offers two practical advantages: (i) Composable with any existing deterministic tie-breakint rule (ii) Neither hard nor soft fork required for deployment. As a representative case, we evaluate stubborn mining—Equal-fork, Lead, and Trial— a class of attacks that leverage long-lived forks, using the blockchain simulator SimBlock(600s blocks generation interval; 100 miners; 0s and 6s latencies). With CA+Smallest-Hash, the attacker profit threshold(PT) rises from 0.2548 $\rightarrow$ 0.2868 (Equal-fork) and 0.2273 $\rightarrow$ 0.2528 (Trial), and Lead becomes unprofitable when $\alpha \leq 0.4$. Under CA+PPoW at 6s latency, PT improves from $\approx 0.255 \rightarrow \approx 0.312$(Trial). These results show that CA systematically reduces attacker revenue and stabilizes fork choice while preserving compatibility.

Keywords: Blockchain, Tie-breaking Rule, Stubborn Mining

1. Introduction

Blockchain technology provides the basic infrastructure for decentralized systems, allowing secure transactions between participants who do not trust each other, without relying on a central authority. In early systems such as Bitcoin [1], consensus is reached through a Proof-of-Work (PoW) mechanism, where miners compete to solve computational puzzles. The network follows the longest-chain rule, meaning that the chain with the greatest total computational work is treated as the valid record.

However, the security of the longest-chain rule is not absolute. Attackers can take advantage of the protocol's incentives through methods like selfish mining [2], in which blocks are deliberately withheld and released at specific times to create forks. This forces honest miners to waste resources on branches that are later discarded, enabling the attacker to gain a larger share of the rewards. Stubborn mining [3], a more aggressive form of selfish mining, keeps extending forks under certain conditions to further increase profits. This idea of keeping a forked chain active is also used in other attacks, such as the Balance Attack, which leverages network delays to maintain the fork and disrupt consensus.

To reduce these risks, different tie-breaking rules have been proposed, especially to defend against selfish mining. Examples include random choice [2], picking the one whose block has the smallest hash value, or using a "last-generated" rule based on timestamps or other criteria. Most of these rules decide based on properties of the fork's tip — the most recent block in each chain [4].

This raises the question of how effective existing tie-breaking rules are against long-fork attacks represented by stubborn mining. In this paper, we evaluate the performance of several current tie-breaking rules in PoW blockchains under such attacks. To enhance their defense, we propose and evaluate a new tie-breaking property called the "common ancestor," which bases decisions on the fork's origin rather than its tip, aiming to improve protection against long-fork attacks like stubborn mining

2. Related Work

A common method to resolve blockchain forks is the Random Choice Rule, introduced by Eyal et al. in their study of Selfish Mining. This rule breaks ties by randomly selecting one competing chain, preventing any party—including attackers with superior network power—from deterministically controlling the outcome. Each chain has an equal

[1] Kyoto University
Sakyo-ku, Kyoto 606-8501, Japan
[a)] xu.zibo.58p@st.kyoto-u.ac.jp
[b)] sakurai.akira.55t@st.kyoto-u.ac.jp
[c)] distributed.nakai@gmail.com
[d)] shudo@media.kyoto-u.ac.jp

chance of being chosen, which reduces certain attack risks. This rule is used by platforms like Ethereum[6].

The Last-Generated Rule offers a deterministic alternative by selecting the most recently created block. Heilman implemented this by embedding trusted timestamps into blocks [5]. However, reliance on a trusted timestamp authority risks centralization and potential malicious behavior, conflicting with blockchain's trustless design.

To overcome this, Lee et al. proposed using the average timestamp of transactions within a block as a decentralized measure [4]. But this can be manipulated by attackers submitting artificially recent transactions and may cause performance overhead due to mandatory timestamping.

Sakurai et al. further advanced tie-breaking with Partial Proof of Work (PPoW), using lower-difficulty block headers as fine-grained time markers [7]. This method resists selfish mining without requiring protocol changes or trusted parties. It shows improved defense while maintaining low overhead and compatibility with existing systems like Bitcoin.

In particular, these rules have been evaluated mainly against selfish mining, showing varying effectiveness. However, their performance against variants such as stubborn mining remains unexplored. In addition,all focus on tie decisions based on chain tips. This paper introduces a novel property that may improve this limitation. The following sections will briefly introduce stubborn mining and investigate the performance of the rules against it to evaluate whether combining these rules with the new property improves their defense.

3. Stubborn Mining

Building upon the principles of selfish mining, Nayak et al. introduced a more generalized and often more profitable class of strategies known as stubborn mining [3]. The core insight is that an attacker can increase their relative revenue by being more persistent, or "stubborn," in mining their private chain, even under circumstances where a traditional selfish miner would concede and adopt the public chain. This approach expands the strategic options for a malicious miner aiming to maximize their rewards beyond their proportional hash power.

These strategies are modeled using a state machine where the state represents the attacker's advantage as Figure 1. The state is primarily defined by the length difference between the attacker's private chain and the public chain, and by whether a fork exists. For example, the `state = 1` signifies that the attacker's private chain is one block ahead without a fork. In contrast, the state `state = 1'` also indicates a one-block lead for the attacker, but a fork has occurred, causing the honest miners' hash power to be split between the two competing chains.

This stubborn behavior is applied through several different strategies, which can be used alone or in combination:

- **Lead-Stubborn (L-stubborn):** When an attacker's private chain has a lead of $k \geq 2$ and an honest miner

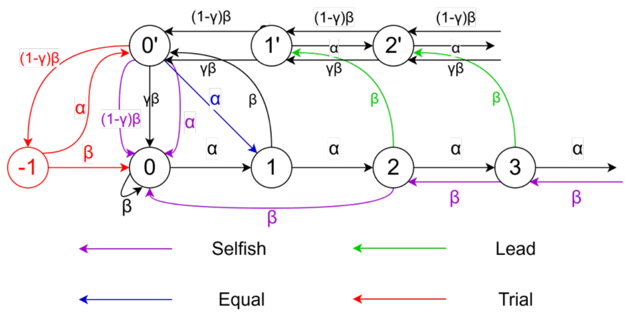

Fig. 1 State transition diagram of the stubborn mining strategy.

finds a new block, the attacker reveals one block to create a tie. This causes a state transition from `state = k` to `state = (k-1)'`. This action creates a fork, forcing a fraction of the honest network to waste hash power on a chain that the attacker intends to obsolete.
- **Equal Fork-Stubborn (F-stubborn):** In the event of a tie (`state = 0'`), if the attacker finds the next block, they conceal it and continue mining privately. This action results in a state transition to `state = 1`, where the attacker now has a one-block advantage.
- **Trial-Stubborn (T_j-stubborn):** This strategy involves the attacker continuing to mine their private chain even after he has fallen behind the public chain. The attacker only gives up when they are $j+1$ blocks behind (a transition from `state = -j` to `state = 0` if an honest miner finds a block). If the attacker successfully catches up to a one-block deficit, the state transitions to `state = 0'`, where all honest miners are still on the other fork.

These strategies can be combined into hybrid approaches, demonstrating that the optimal mining strategy for an attacker is not a "one-size-fits-all" solution but depends heavily on network parameters such as the honest miners' hash power (β), the attacker's hash power (α = 1 - β), and the attacker's advantage in network propagation (γ).

4. Proposed Method

We introduce a novel tie-breaking property called the Common Ancestor (CA). Unlike most existing rules that determine a winner based on the volatile properties of competing chains' tips, our core idea is to shift the basis for tie resolution to the stable and immutable block where the fork began. Specifically, when competing chains emerge, this method traces back to their last shared block—the "common ancestor." The winner is then determined based on a deterministic property of this ancestor block itself, or its relationship with the divergent child blocks. We refer to this principle as "reusing tie outcomes." This method uses the block where the fork started to break a tie. Since that block is part of the permanent history, the tie is always broken the same way. This makes the network more stable and avoids chain reorgs.

A significant advantage of the CA property is its compatibility and non-invasive nature. It is not intended to replace

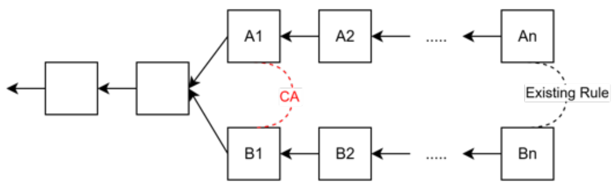

Fig. 2 Common Ancestor under 2 fork.

existing tie-breaking rules (such as selecting the lowest hash or earliest timestamp) but rather to act as a ”meta-rule” that dictates where to apply them. Therefore, it can, in principle, be combined with the majority of existing deterministic tie-breaking rules, enabling the enhancement of major blockchain systems like Bitcoin and Ethereum without altering their core logic. More importantly, implementing the CA property does not require a hard or soft fork. It is essentially a logical optimization in the client software for choosing the main chain, not a modification of the consensus rules themselves. Nodes can independently adopt this more robust logic. Because the rule is deterministic, all nodes that adopt it will independently and consistently converge on the same chain, obviating the need for a disruptive network upgrade while effectively strengthening consensus stability and significantly reducing the risk of reorgs caused by tie-breaking ambiguity.

Algorithm 1 Common Ancestor Tie-Breaking

```
1: procedure CATIE($block_1, block_2, DeterministicRule$)
2:     $b_{original\_1} \leftarrow block_1$
3:     $b_{original\_2} \leftarrow block_2$
4:     $temp_1 \leftarrow block_1$
5:     $temp_2 \leftarrow block_2$
6:     while $block_1 \neq block_2$ and $block_1 \neq$ NIL and $block_2 \neq$ NIL do
7:         if $block_1$.height $\geq$ $block_2$.height then
8:             $temp_1 \leftarrow block_1$
9:             $block_1 \leftarrow block_1$.prevBlock
10:        else
11:            $temp_2 \leftarrow block_2$
12:            $block_2 \leftarrow block_2$.prevBlock
13:        end if
14:    end while
15:    $winner \leftarrow DeterministicRule(temp_1, temp_2)$
16:    if $winner = temp_1$ then
17:        return $b_{original\_1}$
18:    else
19:        return $b_{original\_2}$
20:    end if
21: end procedure
```

Let’s take Figure 2 as an example, this scenario involves two forks. If the lengths of chain A and chain B are identical, a tie occurs. Under a deterministic tie-breaking rule, honest miners determine the winner based on certain deterministic properties of An and Bn, namely the tips of chain A and chain B, and then decide which chain to continue mining on. In contrast, under CA, the decision is made according to the properties of A1 and B1. The specific implementation is shown in **Algorithm 1**. Hereafter, we will provide a more detailed discussion based on several specific rules.

5. Evaluation

This section presents a detailed evaluation of several specific tie-breaking rules within the framework of PoW blockchains that utilize the longest-chain rule. Our primary objective is to assess the efficacy of these rules in mitigating the Stubborn Mining attack, analyzing scenarios both with and without the implementation of the CA property. To facilitate a consistent comparison, we employ a unified evaluation metric. This metric will be used to quantify how the CA property impacts the performance of malicious strategies that aim to maintain forks for disproportionate reward gains. It is important to note that while the theoretically optimal Stubborn Mining strategy involves a sophisticated combination of three basic strategies, our simulation, for the sake of analytical clarity, evaluates each basic strategy in isolation. The rationale behind this methodological simplification is our hypothesis that if the CA property can effectively suppress each strategy independently, it will logically extend its suppressive effects to any combination of these strategies.

5.1 Simulation Setup

We use SimBlock to simulate the blockchain network [8]. The block generation time is set to 600 seconds, consistent with Bitcoin. To better evaluate the performance of the CA property under different latencies, we conduct comparative experiments with delays of 0 and 6 seconds. Block propagation among honest miners is assumed to have uniform delay. For each experiment, α ranges from 0.05 to 0.4, allowing us to observe the impact of CA on attackers with different hash powers under the same latency. To reflect the attacker’s advantage, the attacker experiences zero delay in both receiving and propagating blocks.

We simulate a network of 100 miners, each holding an equal share of the total mining power. When the attacker controls a fraction α of the total power, it effectively controls $100{\cdot}\alpha$ miners, which form a pool to carry out Stubborn Mining.

For each simulation, we set the termination condition as reaching 100,000 blocks on the main chain. Let M_i denote the number of blocks mined on the main chain by miner i; it is straightforward that the sum of all M_i equals 100,000. Based on this, we define the revenue of attacker as in Equation 1.

$$R_{attacker} = \sum_{i=1}^{100\cdot\alpha} M_i/100{,}000 \qquad (1)$$

Ideally, an honest miner’s revenue is proportional to its hash power, an attack is considered profitable only if the attacker’s revenue exceeds its hash power. Accordingly, we define the profit threshold(PT) as in Equation 2.

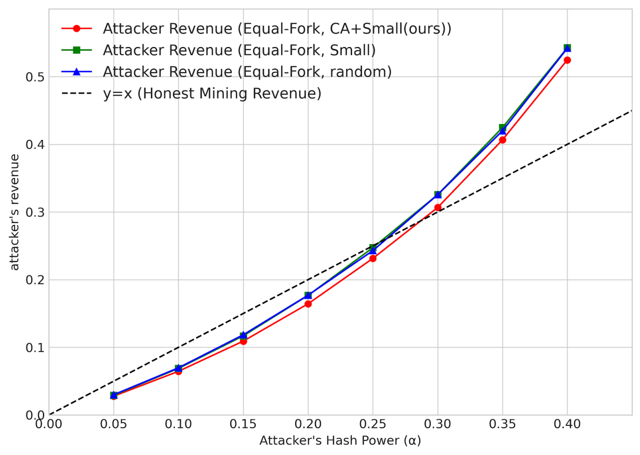

Fig. 3 Equal Fork under 0 delay

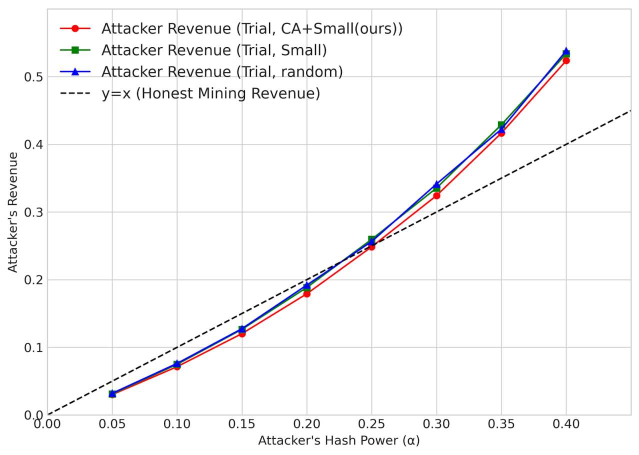

Fig. 5 Trial under 0 delay

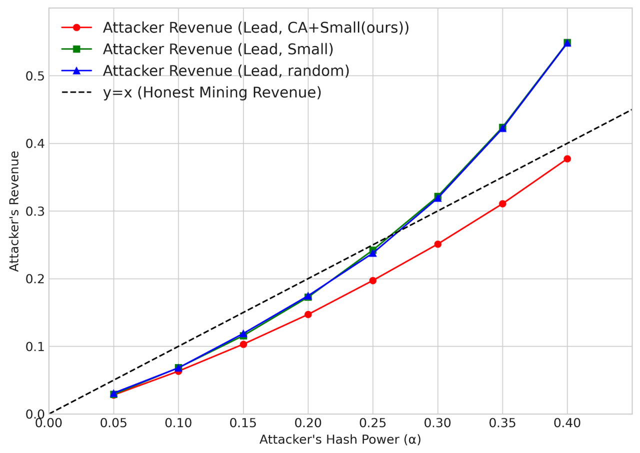

Fig. 4 Lead under 0 delay

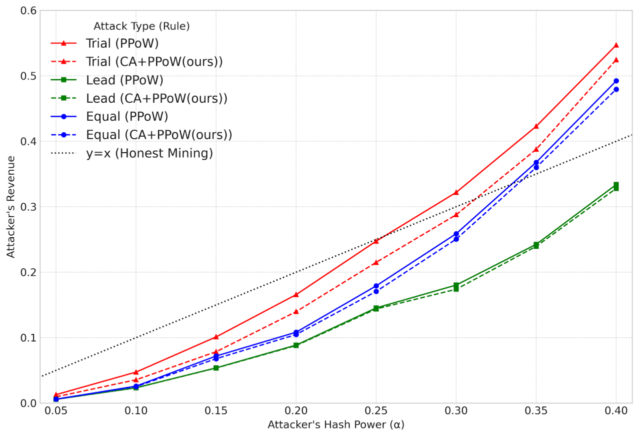

Fig. 6 ppow tie-breaking rule under 6s

$$\mathrm{PT} = \min(\alpha), \quad \text{when } R_{\text{attacker}} \geq \alpha \qquad (2)$$

For the choice of deterministic tie-breaking rules, we select the Smallest Hash rule and the Last-Generated rule. This is because the former was the official tie-breaking rule in Bitcoin systems, while the latter is generally effective in mitigating Selfish Mining. In the case of the Last-Generated rule, although there are several variants, we adopt the most recent PPoW-based Last-Generated rule in this simulation.

5.2 Result

Figures 3–5 present the attacker's revenue under the Equal-fork, Lead, and Trial attacks. A careful comparison of these results shows that the `CA+Small` rule, which combines the CA property with the Smallest Hash rule, consistently provides stronger protection than the standard `Small` rule. This difference can be observed across all three attack strategies. In the Equal-fork attack, the PT under `Small` is $\alpha = 0.2548$, whereas under `CA+Small` it increases to $\alpha = 0.2868$, giving the attacker a noticeably narrower range of profitability. In the Trial attack, a similar effect is observed: the threshold rises from $\alpha = 0.2273$ to $\alpha = 0.2528$, meaning that attackers require a larger fraction of computational power before the attack becomes worthwhile. The most striking case appears in the Lead attack. Here, the `Small` rule still permits profitability at $\alpha = 0.2633$, but with `CA+Small`, no profitability is observed at all within our tested range ($\alpha \leq 0.4$). This indicates that the CA property not only pushes profitability thresholds higher, but in some scenarios it can suppress the attack so effectively that profitability disappears entirely. By looking at these three attacks together, it becomes clear that `CA+Small` provides a more reliable line of defense, and that the CA property is a key element in strengthening consensus security.

Following this, Figure 6 extends the analysis by comparing the revenue curves of the three attacks under the `PPoW` and `CA+PPoW` rules. The relationship here closely mirrors what we observed with `CA+Small` and `Small`: the rule enhanced with the CA property, `CA+PPoW`, consistently raises the PT compared with the baseline `PPoW`. For the Trial attack, the improvement is particularly strong, with the threshold increasing from about $\alpha = 0.255$ under `PPoW` to about $\alpha = 0.312$ under `CA+PPoW`. This corresponds to a relative increase of more than 22%, which makes the attack substantially harder to carry out. The Equal-fork attack also shows a positive effect, though on a smaller scale, with the threshold shifting from $\alpha \approx 0.335$ to $\alpha \approx 0.342$. The Lead attack provides a different but still meaningful picture. In this case, both `PPoW` and `CA+PPoW` keep the revenue curve below the honest-mining profitability line for $\alpha \leq 0.4$, but a closer look at the curves reveals that `CA+PPoW` is consistently lower than `PPoW` across this range. Although our simulations do not extend beyond $\alpha = 0.4$, this trend strongly suggests that the profitability threshold under `CA+PPoW` must be higher than under `PPoW`, even if the exact point cannot be determined within the tested range.

6. Conclusion

In this paper, we proposed the Common Ancestor (CA) property as a new tie-breaking property for PoW blockchains. Unlike traditional rules that rely on properties of the competing chains' tips, our approach determines the winner based on the fork's origin, ensuring consistent and stable outcomes. We conducted simulations using SimBlock to compare the performance of existing deterministic rules and their CA-enhanced variants against stubborn mining strategies. The results show that CA-based rules, such as `CA+Small` and `CA+PPoW`, consistently raise the profitability threshold for attackers compared to their baseline counterparts. The degree of improvement varies across different attack strategies, which can be explained by the fact that the more frequently tie outcomes are reused, the more effectively the network converges to consensus, thereby suppressing attacks. In some cases, such as the Lead attack under `CA+Small`, profitability disappears entirely within the tested range, highlighting the strong defense provided by CA. These findings indicate that the CA property significantly strengthens consensus security while requiring no disruptive protocol changes. As a future task, we plan to extend the evaluation to hybrid stubborn mining strategies and investigate the resistance of CA-enhanced rules under more complex network conditions.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number JP24H00691, JST ACT-X Grant Number JPMJAX24M9, and JST BOOST Grant Number JPMJBS2407 .

References

[1] Nakamoto, S.: *Bitcoin: A Peer-to-Peer Electronic Cash System*, White Paper, https://bitcoin.org/bitcoin.pdf (2008).

[2] Eyal, I. and Sirer, E. G.: *Majority is not Enough: Bitcoin Mining is Vulnerable*, arXiv:1311.0243 (2013).

[3] Nayak, K., Kumar, S., Miller, A., and Shi, E.: "Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack," In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 305–320 (2016).

[4] Lee, J. and Kim, Y.: "Preventing Bitcoin Selfish Mining Using Transaction Creation Time," In *2018 International Conference on Software Security and Assurance (ICSSA)*, pp. 19–24 (2018).

[5] Heilman, E.: "One Weird Trick to Stop Selfish Miners: Fresh Bitcoins, a Solution for the Honest Miner (Poster Abstract)," In: *Financial Cryptography and Data Security* (R. Böhme, M. Brenner, T. Moore, and M. Smith, eds.), pp. 161–162, Springer Berlin Heidelberg (2014).

[6] Wood, G.: *Ethereum: A secure decentralised generalised transaction ledger*, Ethereum Project Yellow Paper (2014), https://ethereum.github.io/yellowpaper/paper.pdf.

[7] Sakurai, A. and Shudo, K.: *Tie-Breaking Rule Based on Partial Proof of Work in a Blockchain*, arXiv preprint arXiv:2403.15030 (2024).

[8] Aoki, Y., Otsuki, K., Kaneko, T., Banno, R., and Shudo, K.: "SimBlock: A Blockchain Network Simulator," In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 325–329 (2019).