Implementation and Evaluation of Commit-Reveal Mitigation Technique for Blockchain Extractable Value

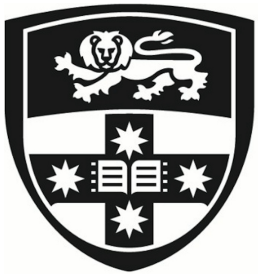
MOHAMMAD SAAD AZAM
SID: 510048152

Supervisor: Dr. Vincent Gramoli

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Bachelor of Advanced Computing (Honours)

School of Computer Science
The University of Sydney
Australia

16 November 2024



THE UNIVERSITY OF
SYDNEY

Student Plagiarism: Compliance Statement

I Mohammad Saad Azam certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that it is not my own by acknowledging the source of that part or those parts of the work.

Name: Mohammad Saad Azam

Signature:  **Date**:14/11/2024

Abstract

Blockchain extractable value (BEV), a critical vulnerability in decentralized finance (DeFi), occurs when miners or validators exploit transaction ordering to extract value. This study empirically evaluates a novel BEV mitigation technique that leverages unbiased randomness in transaction ordering. Our modular and adaptable approach, designed for seamless integration across blockchain networks, utilizes partial seeds from a leader set to produce secure randomness. By employing a pseudo-random generator (PRG) to generate transaction permutations and implementing transaction chunking, this technique achieves a decentralized solution with minimal overhead while preserving the original blockchain's consensus properties.

Our hypothesis posits that this randomness-based approach can mitigate BEV effectively without sacrificing computational efficiency or system security. To test this, we deployed a controlled Ethereum network, using UniSwapV2 contracts to simulate common DeFi vulnerabilities like sandwich attacks and arbitrage. Custom contracts were also implemented to simulate other BEV scenarios, along with a sandwich bot to model a "rushing adversary," the most prevalent BEV exploit in DeFi.

Real-time data from Etherscan for UniSwap over a three-week period was used to assess the technique across metrics such as BEV reduction, computational efficiency, decentralization, security, and throughput. Results showed a substantial BEV reduction as hypothesized, indicating that this approach may outperform other solutions *on certain criteria*. Importantly, our findings demonstrate that this technique can be overlaid on existing blockchain infrastructures, although requiring considerable ecosystem modifications. This suggests a robust, decentralized, and low-overhead strategy for BEV mitigation, but comes at the cost of delay in finalization.

Acknowledgements

I would like to thank Vincent for his unwavering patience, as well as valuable insights. I would also like to thank Andrei for setting up my VMs and Pouriya for introducing me to the research. My parents and siblings for making sure I didn't slack off, and finally, my friends who had to sit through me talking about the work I had to do and yet never seemed to do.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

DeFi is a subset of decentralized protocols that operates autonomously on blockchain-based smart contracts, specifically aimed at creating a financial system independent of traditional intermediaries. The ecosystem has grown substantially, with over 90 billion USD locked in DeFi platforms [HowDarkIsTheForest], including automated market makers (AMMs), lending platforms, and margin trading systems. Automated Market Maker (AMM) exchanges, like Uniswap, represent a key component of DeFi's infrastructure. Unlike traditional exchanges, which rely on a limit order book of bids and asks, AMMs use liquidity pools governed by smart contracts to facilitate trades. The constant product rule is the most widely used mechanism in these exchanges; in a two-asset pool, the product of the assets' quantities remains constant, establishing a predictable pricing model.

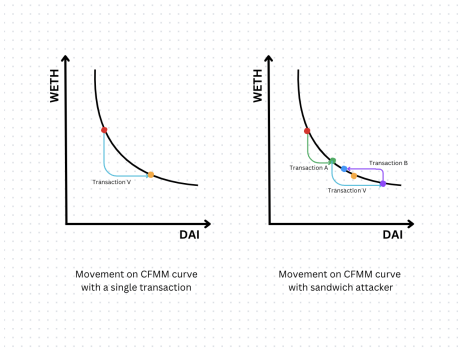


FIGURE 1.1. Slippage on CFMM Curve: The left chart shows the expected slippage for a single transaction on the Constant Function Market Maker (CFMM) curve. The right chart illustrates unexpected slippage introduced by a sandwich attack

However, trading on AMMs exposes users to slippage, where the actual execution price differs from the expected price due to the available liquidity in the pool and market volatility. As trades increase in size, prices tend to shift unfavorably; a phenomenon called expected slippage. Additionally, validators or specialized agents can exploit transaction ordering to manipulate the execution price, creating unexpected slippage. In response, traders often set a slippage tolerance to limit these adverse price changes, yet this

does not entirely prevent exploitation through MEV attacks. In response to the pervasive threat of BEV, researchers have proposed various countermeasures, broadly categorized into four primary approaches:

Time-based order fairness protocols enforce a first-come-first-served structure within consensus mechanisms by ordering transactions based on the timing of submission. In this model, if a majority of validators receive transaction T before T', T' should appear in the block no later than T. Known as receive-order fairness and first proposed by Kelkar et al., this approach relies on committee-based trust assumptions, typically allowing up to f malicious nodes within a network. However, this dependency on honest majorities, along with significant communication overhead, limits the practicality of time-based fairness protocols in high-throughput environments like DeFi, where transaction volume and adversarial behavior are prevalent. This setup makes such protocols susceptible to network congestion and censorship

Content-agnostic ordering, often implemented through commit-and-reveal protocols, establishes transaction order independently of transaction content to reduce front-running opportunities. In this approach, users initially send commitments with minimal metadata, like transaction fees, allowing miners to sequence transactions without knowing their content. After the ordering is set, the transactions are revealed and executed. This method is commonly realized with cryptographic tools like threshold encryption, where a committee decrypts the transactions only after the order is established, or time-lock encryption, which schedules decryption to prevent early disclosure. However, content-agnostic ordering has notable limitations: while effective at concealing transaction details initially, it often suffers from metadata leakage, potentially revealing sender information. Additionally, these methods require significant trust in the committee's integrity or the precision of time-lock mechanisms, making them vulnerable to early decryption by malicious actors and challenging to implement without risking security or network efficiency.

External Fair Ordering Mechanisms: Classic approaches, like Flashbots or Eden, introduce trusted third parties to bundle and order transactions, reducing adversaries' control over transaction positioning. However, these solutions often introduce dependencies on external parties or reduce protocol efficiency.

In this work, I implement and evaluate a different approach: the **Partitioned and Permuted Protocol (P3)** [EatingSandwiches]. This protocol leverages *on-chain randomness* to counteract front-running

without relying on external resources, thus preserving both efficiency and decentralization. By increasing the state space through transaction chunking and introducing unbiased random permutations in transaction ordering, P3 effectively renders sandwich attacks unprofitable. I have adapted and implemented this protocol on Ethereum, assessing its potential to integrate seamlessly with existing blockchain infrastructure while minimizing MEV opportunities.

The contributions of this study are as follows:

Protocol Adaptation and Integration: The protocol was adapted to operate within the Ethereum environment using a modified Geth Execution client. Done mainly through a smart contract, which extracts commitments from a leader set, generating the seeds required for transaction permutation, thereby embedding the protocol's core mechanics directly on-chain.

Chunking and Reordering Implementation: transaction chunking and reordering techniques were implemented to align with the protocol's design. This was achieved through smart contracts that divided transactions into multiple chunks, allowing for enhanced randomization in their execution order. These mechanisms were incorporated into Ethereum's transaction processing and block building.

Simulation and Real-World Data Testing: To assess the protocol's effectiveness under realistic conditions, I deployed a dedicated Ethereum node that replayed historical Etherscan transactions from the past two weeks. In parallel, a custom sandwich bot was introduced to submit MEV attack transactions, replicating adversarial behavior commonly observed in decentralized finance.

CHAPTER 2

Background

2.1 Decentralized Finance (DeFi)

Decentralized Finance (DeFi) refers to a blockchain-based financial infrastructure that has rapidly gained traction in recent years. It encompasses a stack of open, permissionless, and highly interoperable protocols built on public smart contract platforms, such as Ethereum (Buterin, 2023). DeFi replicates traditional financial services in a more open and transparent manner, eliminating intermediaries and centralized institutions. Instead, DeFi relies on decentralized applications (DApps) and open protocols, where agreements are enforced through code and transactions are executed in a secure, verifiable manner. These state changes persist immutably on a public blockchain, resulting in a financial system characterized by unprecedented transparency, equal access rights, and a reduced need for traditional custodians, clearing houses, or escrow services. In DeFi, these roles are largely replaced by "smart contracts" (Schär, 2021).

DeFi already supports a wide range of applications. For instance, users can acquire USD-pegged stablecoins on decentralized exchanges (DEXes), lend these assets on decentralized lending platforms to earn interest, and subsequently allocate interest-bearing instruments to decentralized liquidity pools or on-chain investment funds.

The backbone of DeFi lies in smart contracts, small programs stored on a blockchain and executed in parallel by a large number of validators. These contracts enable participants to verify the correct execution of operations independently. Although public blockchains and their smart contracts are less efficient compared to traditional centralized systems, they offer unparalleled transparency and security. Properly implemented smart contracts guarantee tamper-proof execution, significantly reducing the risk of manipulation or arbitrary intervention (Schär, 2021).

2.2 Gas and Fees in Ethereum

An Ethereum transaction can either transfer funds to a non-executing account address or send input data (and optionally funds) to a smart contract address. Transactions are broadcast across the peer-to-peer Ethereum network, signaling their availability for inclusion in a mined block. These transactions can exist in one of three states: unconfirmed and not yet mined, confirmed and executed, or rejected as invalid by the Ethereum network (Buterin, 2023).

The execution of Ethereum transactions consumes gas, which acts as a pseudo-currency representing the computational effort required by a miner (and other nodes) to process the transaction. Smart contracts on Ethereum can include complex logic, loops, and conditions, making their exact gas consumption determinable only through execution. Ethereum meters gas consumption via a fixed mapping of contract opcodes to gas units. This ensures a transparent and predictable cost model for executing transactions (Buterin, 2023).

2.3 Blockchain Extractable Value (BEV) and its Exploitation

The transparent nature of DeFi has facilitated its rapid adoption but has also made it susceptible to exploitation. With over $90 billion locked in DeFi protocols, distributed ledgers have demonstrated their ability to mediate trustlessly among financial actors exchanging hundreds of millions of dollars daily (Qin et al., 2022). However, this transparency allows malicious actors to exploit the publicly visible nature of transactions. Miners, for instance, can control the ordering of transactions in their mined blocks, creating an information asymmetry that can be leveraged for financial gain.

In addition to miners, predatory traders have emerged, specializing in extracting revenue through a range of manipulation techniques, such as high-frequency attacks, pump-and-dump schemes, and wash trading. Automated trading bots monitor the mempool for profitable transactions and insert their own with higher gas fees, ensuring they are executed first. This behavior, referred to as Blockchain Extractable Value (BEV) or Miner Extractable Value (MEV), represents the profit that miners or other participants can extract by manipulating transaction ordering, inclusion, or exclusion (Qin et al., 2022).

2.4 MEV and Transaction Ordering Taxonomy

The order of transactions within a block significantly impacts blockchain value extraction. Recent research introduces a taxonomy of transaction ordering, categorizing manipulative behaviors into destructive front-running, tolerating front-running, and back-running (Qin et al., 2022):

- **Destructive Front-Running:** In this case, an adversary's transaction (T_A) precedes the victim's transaction (T_V), causing T_V to fail. The adversary exploits the opportunity without regard for the consequences on other transactions, such as state reversion in the victim's operation.
- **Tolerating Front-Running:** The adversary ensures that T_V executes successfully, which is essential for specific attack strategies like sandwich attacks. Here, the adversary profits by creating favorable conditions for T_A without disrupting T_V.
- **Back-Running:** The adversary places T_A after T_V, profiting from the state changes caused by T_V. For example, back-running can be effective in scenarios involving oracle updates or the trailing phase of a sandwich attack. Notably, back-running is often less costly than front-running since it avoids competitive fee bidding.

These forms of transaction manipulation exploit the inherent transparency of DeFi systems and their reliance on a deterministic transaction ordering process. By leveraging techniques such as front-running, adversaries can extract value from ordinary user transactions, amplifying the systemic risks associated with MEV (Qin et al., 2022).

2.5 MEV and Systemic Risks

While blockchains and smart contracts promise fair and transparent trading ecosystems, this promise has not been fully realized. Research has documented the widespread deployment of arbitrage bots in blockchain systems, particularly in DEXes. These bots exploit inefficiencies by leveraging high transaction fees and low network latency to front-run ordinary users' trades. The study in (Daian et al., 2020) quantifies the scope of such arbitrage strategies using real-world blockchain data and introduces the concept of Priority Gas Auctions (PGAs). In a PGA, bots competitively bid up transaction fees to secure early block positions, thereby ensuring their transactions are executed ahead of others.

The analysis in (Daian et al., 2020) highlights the risks associated with these practices, particularly their impact on blockchain consensus-layer security. High fees paid for priority transaction ordering pose a measurable risk to the network, as these costs incentivize miners to reorder transactions in ways that may undermine the integrity of the system. Moreover, the study emphasizes that MEV is not limited to DEXes but represents a broader class of challenges affecting the entire blockchain ecosystem. These findings underscore the importance of addressing MEV as a systemic issue to safeguard the fairness, security, and long-term sustainability of blockchain networks.

CHAPTER 3

Literature Review

3.1 Types of MEV Attacks

Decentralized Finance (DeFi) and Decentralized Exchanges (DEXs) have revolutionized the financial landscape by enabling peer-to-peer transactions and removing the need for centralized authorities. Built on blockchain technology, these platforms leverage smart contracts to ensure transparency, security, and accessibility in financial transactions. However, the very features that make DeFi innovative also introduce significant vulnerabilities. One such critical issue is the emergence of Maximal Extractable Value (MEV), which refers to the value that can be extracted by reordering, including, or censoring transactions within a block. This section explores the various types of MEV attacks that exploit these vulnerabilities.

DEXs have inherent design flaws that allow arbitrage bots to exploit market inefficiencies, mirroring strategies seen in traditional financial markets. First discovered and published in [5], these bots engage in behaviors such as front-running, aggressive latency optimization, and priority gas auctions (PGAs)(This paper is so influential that the number of bots in the network had increased drastically after the draft was published). PGAs occur when bots competitively bid up transaction fees to secure favorable positions in a block, ensuring their transactions are executed ahead of others (Daian et al., 2020). This competitive behavior stems from the pure revenue opportunities present in DEXs, where atomic trades guarantee unconditional profits across multiple assets. Research has identified a minimum economy of $6M arising from such opportunities, highlighting the scale of exploitation.

(Daian et al., 2020) evaluates how MEV also incentivizes miners to reorder transactions to extract additional value, leading to what are termed ordering optimization (OO) fees. These fees have systemic implications, as they can encourage fee-based forking attacks, where miners alter blockchains to maximize

profit. These attacks highlight how MEV exploitation can threaten the fundamental consensus-layer security of blockchain systems.

First formalized and evaluated in (Qin et al., 2022), Front-running, a prevalent MEV attack, occurs when an adversary observes a pending transaction in the mempool and inserts their transaction ahead of it by bidding a higher gas fee. This tactic ensures their transaction is processed first, often at the expense of the victim. A specific and damaging type of front-running is the sandwich attack, where an adversary places a buy transaction before and a sell transaction after a victim's trade. This manipulation exploits slippage and price changes, enabling the attacker to profit from the victim's transaction. Empirical analysis has quantified the prevalence of sandwich attacks. Over a period of 32 months, researchers identified 750,529 sandwich attacks on Uniswap, Sushiswap, and Bancor, yielding a total profit of $174.34M. Interestingly, many of these attacks—31.98%—were privately relayed to miners using zero-gas-price transactions via MEV relay systems. This emphasizes the evolving sophistication of MEV bots and their reliance on private infrastructure to maximize extraction. (Zhou et al., 2021) goes one step further and deploys a sandwich bot on the mainnet, and they show that they were able to generate revenues of several thousands of dollars daily. (Angeris et al., 2023) introduces a theoretical 'Cost of MEV' using fairness functions.

While front-running has received significant attention, back-running, where an attacker positions their transaction after a target transaction, is another method of MEV extraction which is evaluated in (Daian et al., 2020). For instance, back-running can be used in sandwich attacks to close positions opened by front-running transactions, ensuring profit from price changes. Cancellation griefing, another related attack, involves an adversary preempting a pending transaction cancellation by issuing a fill order with a higher gas price, thereby exploiting the order before it can be canceled. This attack highlights the challenges faced by partially centralized exchanges, where orders are often visible to adversaries in the mempool.

(Eskandari et al., 2020) looks at MEV attacks beyond the DeFi in applications such as crypto-collectibles and gambling. For example, in games like Cryptokitties, adversaries can front-run calls and block legitimate user transactions (fatal-front running of sorts). In gambling applications like Fomo3D, attackers use gas auctions to suppress competing transactions, ensuring their advantage in time-sensitive scenarios - an example of 'clogging;. These examples demonstrate the breadth of MEV exploitation across diverse decentralized applications.

(Choi and Kim, 2024) studies the prevalence of MEV attacks in DEXs underscores their systemic nature. Analysis of transaction data from Uniswap's USDC/WETH pool reveals that MEV bots account for approximately 45% of the total daily average trading volume, amounting to $480M out of $1B. This highlights the significant financial impact of MEV attacks on DeFi ecosystems and underscores the need for robust mitigation strategies.

(Daian et al., 2020), again, studies and evaluates MEV attacks, particularly those involving arbitrage bots and sandwiching, not only affect individual traders but also pose systemic risks to blockchain networks. By incentivizing manipulative behaviors such as PGAs and fee-based forking attacks, MEV undermines the integrity and fairness of transaction ordering. These vulnerabilities threaten the security and stability of DeFi platforms, necessitating comprehensive solutions to address both the root causes and consequences of MEV exploitation.

3.2 Consensus-Based Fair Ordering

Consensus-based fair ordering protocols have emerged as potential solutions to mitigate transaction-ordering attacks, such as front-running and back-running, which have significantly impacted the integrity and fairness of decentralized systems. By incorporating fairness guarantees directly into the consensus mechanism, these protocols aim to prevent adversarial reordering of transactions. However, despite notable advancements, several limitations persist, leaving room for further improvement.

Pompe (Zhang et al., 2020) introduces Byzantine ordered consensus, which augments traditional Byzantine Fault Tolerant (BFT) State Machine Replication (SMR) with guarantees on the specific order of commands. This mechanism aims to prevent Byzantine nodes from exploiting ordering decisions, thereby addressing concerns over transaction manipulation. Pompe ensures that transaction ordering respects a natural extension of linearizability, offering an absolute and deterministic ordering mechanism. However, Pompe is limited in its ability to defend against rushing adversaries, who exploit network latencies and mempool visibility to insert malicious transactions before the ordering is finalized. As a result, while Pompe provides significant protections against Byzantine oligarchies, its defense against dynamic adversaries remains incomplete.

Themis (Keller et al., 2019) builds upon BFT consensus protocols to enforce strong notions of fair ordering. It integrates fair ordering into its consensus protocols with minimal performance overhead, offering practical protection against transaction-ordering attacks, such as front-running and back-running. While

Themis demonstrates resilience to adversarial manipulation in experimental evaluations, it does not fully eliminate the issue of transaction manipulation. Adversaries can still exploit transaction payload visibility before the ordering is finalized, leaving certain vulnerabilities unaddressed.

Lyra (Zarbafian and Gramoli, 2023) combines a commit-reveal mechanism to obfuscate transaction payloads with a leaderless ordered consensus protocol that predicts transaction order. By eliminating the role of a leader, Lyra aims to prevent reordering attacks often facilitated by Byzantine leaders.

3.3 Mitigation Techniques Using Third-Party Ordering

Mitigation techniques that delegate transaction ordering to trusted third parties, such as Flashbots (Daian et al., 2022), have emerged as a practical solution to address the challenges posed by Miner Extractable Value (MEV). While these solutions are technically effective and widely adopted, they come at the cost of reduced decentralization, thereby altering one of the foundational principles of blockchain systems. Flashbot uses a private pool where the users submit their transaction off-chain directly to miners. These are called dark pools (Qin et al., 2022) and the transactions in the bundle are not visible to the public mempool until they are executed and included in a block. Transaction reordering is eliminated as the user defines the exact order of transactions in their bundles, reducing arbitrary reordering by miners.

Eden Network (Network, 2024) is another mitigation technique that addresses the challenges posed by Miner Extractable Value (MEV). Similar to Flashbots, Eden focuses on providing mechanisms to reduce harmful transaction reordering and front-running in blockchain systems. However, Eden introduces its own model, which delegates transaction ordering to a network of validators while offering enhanced transaction protection and prioritization mechanisms. This approach, while technically effective, also introduces trade-offs in decentralization and access. Eden employs a private transaction routing mechanism where users can send their transactions through a private node network. This network ensures that user transactions are not visible in the public mempool, reducing the risk of front-running, sandwich attacks, or other MEV-related exploits.

Ethereum's Proposer-Builder Separation (PBS) model is a transaction ordering mechanism introduced as part of Ethereum 2.0 to mitigate MEV (Maximal Extractable Value) exploitation and enhance fairness in transaction sequencing. PBS separates the roles of block proposers and block builders, enabling proposers to delegate block construction to specialized builders. This separation is designed to reduce

the influence of proposers on transaction ordering and mitigate the risks associated with MEV-driven transaction manipulation.

In PBS (Foundation, 2024), builders compete in an off-chain auction to construct the most profitable block while proposers select the block offering the highest reward. This framework aims to decouple block construction from transaction ordering, thus limiting the ability of individual proposers to manipulate transaction sequences for personal gain. Additionally, PBS introduces a degree of randomness in proposer selection, further reducing the likelihood of coordinated adversarial behavior. Despite its significant advancements, PBS is not without limitations. The model relies on the assumption that builders act honestly in maximizing block profitability without engaging in harmful behaviors, such as censorship or collusion. Moreover, while PBS minimizes the influence of individual proposers, it does not fully eliminate MEV extraction, as builders themselves can exploit their position to reorder transactions within blocks. Furthermore, the system's reliance on off-chain auctions introduces potential vulnerabilities in ensuring transparency and fairness in builder-proposer interactions.

3.4 Mitigation Techniques Using Committee Ordering

Hashgraph (Hedera, 2016) introduces a novel consensus mechanism that leverages a gossip protocol to achieve fair ordering of transactions. In this approach, committee members exchange information about the transactions they have received in a process known as "gossip about gossip." This mechanism builds a distributed record of the transaction history, allowing members to utilize virtual voting to agree on a consensus order. By replacing traditional voting methods with virtual voting, Hashgraph significantly reduces the complexity of the system while maintaining efficiency and scalability. The primary goal of Hashgraph is to ensure that no small group of attackers can disproportionately influence the order of transactions, thus promoting fairness in transaction processing. However, the definition of fairness in Hashgraph remains somewhat ambiguous, raising questions about the extent to which the protocol guarantees unbiased transaction ordering. This lack of clarity introduces uncertainty about its resilience to sophisticated adversarial strategies.

Despite these concerns, Hashgraph demonstrates significant potential as a fair and efficient consensus mechanism, especially in environments where low latency and high throughput are critical. Its reliance

on virtual voting and gossip protocols provides a unique approach to addressing the challenges of transaction ordering in distributed systems, although further analysis is required to fully understand and validate its fairness guarantees.

Wendy (Kursawe, 2020) introduces a fairness-oriented framework designed to address transaction ordering challenges in decentralized trading markets. Developed in response to the risks of financial fraud stemming from manipulated transaction orderings, Wendy focuses on achieving relative order fairness—ensuring that the relative sequence of transactions is unbiased and equitable. Recognizing that absolute fairness in transaction ordering is provably unattainable under certain conditions, Wendy instead offers a practical suite of protocols to implement varying degrees of fairness within blockchain systems. Wendy operates as an auxiliary widget that complements existing blockchains, making it agnostic to the underlying protocol and its specific security assumptions. This modular design allows Wendy to integrate seamlessly with a wide range of blockchain implementations, providing a flexible tool to enforce fairness without requiring fundamental changes to the consensus mechanism. Moreover, Wendy can be selectively applied to a subset of transactions, enabling the coexistence of multiple independent fair markets on the same chain.

Despite its strengths, Wendy's approach does not guarantee fairness in all scenarios and remains susceptible to subtle adversarial strategies. However, its low overhead and adaptability make it a compelling solution for improving fairness in transaction ordering, particularly in decentralized financial markets where such considerations are increasingly critical.

3.5 Commit-and-reveal schemes

The commit-and-reveal scheme is a cryptographic technique designed to ensure fairness and mitigate MEV (Maximal Extractable Value) by obfuscating transaction details until their inclusion in a block is finalized. This mechanism works by splitting the transaction process into two distinct phases: a commitment phase, where users submit cryptographically hashed commitments containing their transaction details, and a reveal phase, where the actual transaction details are disclosed and executed. By delaying the revelation of transaction contents, commit-and-reveal schemes prevent adversaries from observing and exploiting pending transactions in the mempool. The scheme provides strong guarantees against front-running attacks by ensuring that adversaries cannot access transaction information until it is too

late to influence its ordering or execution. Furthermore, it reduces the prevalence of transaction manipulation strategies, such as sandwiching and back-running, by eliminating the window of visibility that adversaries rely on to deploy these tactics.

However, commit-and-reveal mechanisms (Canidio and Danos, 2024), (Zarbafian and Gramoli, 2023) are not without their limitations. The inherent delay between the commitment and reveal phases introduces latency, which can be problematic in time-sensitive applications, such as decentralized exchanges (DEXs), where rapid execution is critical to maintaining market efficiency. Additionally, this delay may lead to increased transaction costs due to failed or stale transactions caused by volatile market conditions during the commitment period. The approach also imposes additional storage and computational overhead on the blockchain, as both commitments and subsequent reveals must be processed and stored.

3.6 Eating sandwiches: Modular and lightweight elimination of transaction reordering attacks

The protocol (Alpos et al., 2024) aims to mitigate sandwich attacks by introducing a transformative construction for blockchain systems. This approach takes an existing blockchain protocol as input and outputs a modified protocol with identical security properties but with the critical difference that sandwich attacks are rendered unprofitable. The proposed solution emphasizes a fully decentralized design, eliminating reliance on trusted third parties or computationally intensive cryptographic primitives. Furthermore, it achieves these objectives with minimal trade-offs, introducing only a linear increase in transaction latency and a negligible computation overhead.

CHAPTER 4

Methodology

While P3 has undergone theoretical validation and security analysis, including a game-theoretic evaluation to demonstrate its stability under rational miner assumptions, it has not yet been tested on a live blockchain environment like Ethereum. This limitation leaves open critical questions about its practical implications in a real-world setting. Our work addresses this gap by implementing and evaluating a functional model of P3 on an Ethereum test network, thereby assessing its performance across a range of real-world metrics that impact DeFi users directly.

To evaluate P3's effectiveness beyond security considerations, our analysis centers on five main criteria:

- **Scope:** This metric evaluates how effectively P3 mitigates various forms of BEV, encompassing front-running, back-running, sandwiching, and other transaction manipulation tactics. We assess whether the protocol sufficiently broadens its reach across multiple DeFi applications, providing a more generalized defense against MEV.
- **Jostling:** Here, we assess the level of competition among traders for block inclusion at favorable positions. By measuring jostling, we gauge whether P3 incites excessive bidding behaviors, comparing it to the default transaction competition observed on Ethereum.
- **Goodput:** We define goodput as the proportion of genuine transactions—those not associated with attacks or defensive tactics—that the blockchain processes per unit of time.
- **Delay:** Transaction delay measures the latency between a trade's submission and execution. We benchmark P3's impact on transaction delay against standard Ethereum conditions.
- **Cost:** Additional costs from P3 could arise either through increased gas fees or through the process of arbitrarily chunking transactions, which may introduce unintended side effects. By assessing these potential cost impacts, we determine whether P3 imposes direct cost increases on users compared to the baseline transaction costs on Ethereum.

- **Decentralization**: This metric evaluates the impact of P3 on the overall decentralization of the blockchain. We assess whether P3's design and operation influence the distribution of power among network participants, specifically comparing any potential shifts away from Ethereum's baseline decentralization, where miners (or validators in a Proof-of-Stake model) are responsible for block construction. This metric helps determine if P3 preserves or diminishes the decentralized nature of transaction processing on Ethereum.

4.1 Network Environment and Attack Simulation

Everything detailed here can be found at this repo: "https://github.com/mohammadsaad3o5/ethereum-MEV"

4.1.1 Test Network Setup

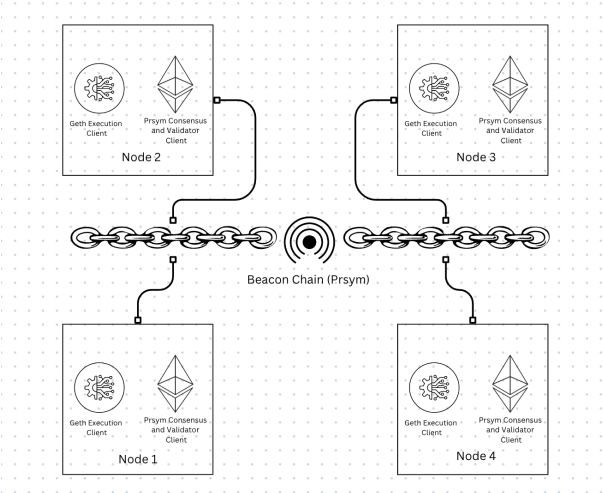


FIGURE 4.1. Diagram of the Test Network Setup: Four interconnected nodes, each running a Geth Execution Client and a Prysm Consensus and Validator Client, are linked through the Beacon Chain (Prysm) to simulate Ethereum's proof-of-stake environment.

The primary purpose of setting up this test network was to create a controlled environment where experimental modifications to the protocol and smart contracts could be deployed and evaluated realistically. This setup allows for direct testing of various decentralized finance (DeFi) mechanisms, including the manipulation of UniSwap reserves and the impact of transaction ordering on vulnerability to sandwich attacks and other transaction manipulation tactics. Additionally, it supports the testing of modified execution client behavior, requiring that all clients adhere to the same protocol rules to prevent consensus

issues. This network configuration is essential for measuring the impact of the implemented Partitioned and Permuted Protocol (P3) and other modifications in a consistent manner.

The network consists of four interconnected nodes, each running a Geth Execution Client (EL) and a Prysm Consensus and Validator Client (CL). These nodes are joined via a Beacon Chain to simulate Ethereum's proof-of-stake environment, ensuring that the consensus and validation processes mimic a real-world blockchain. A customized genesis file was created, pre-funding several accounts, which facilitates controlled testing by providing liquidity and assets to simulate various trading scenarios.

Geth was selected as the execution client due to its popularity, extensive documentation, and robust support for custom modifications. This choice ensures compatibility with Ethereum's infrastructure and ease of customization for our specific protocol requirements.

The expected outcome of this setup is to provide an isolated, reproducible environment where the effects of protocol changes can be assessed independently. By modifying various components, such as transaction ordering and reserve management, this environment enables precise measurement of the protocol's impact on metrics like transaction cost, execution delay, and resistance to MEV (Miner Extractable Value) attacks.

The main limitation of this setup is the absence of network latency, as all nodes run on the same physical machine. However, studies [BlockchainOnBareMetal] indicate that artificially introduced delays can simulate real-world conditions effectively, allowing for accurate insights into latency-related impacts in a live network scenario.

4.1.2 Sandwich Bot Set

	DAI Reserves	WETH Reserves
Bot Transaction (same direction)	R = R + Bot	ROut = ROut - WETH(Bot)
User Transaction	R = R + Bot + user	ROut = ROut - WETH(Bot) - WETH(user)
Bot transaction (opposite direction)	R = R + Bot + user - Bot'	R = R - output

TABLE 4.1. Changes in input and output reserves during a sandwich attack.

The purpose of the sandwich bot within our test network is to simulate and evaluate Miner Extractable Value (MEV) attacks, specifically sandwich attacks, in a controlled environment. This bot acts as a model for a rushing adversary—an entity capable of quickly detecting and reacting to transactions as

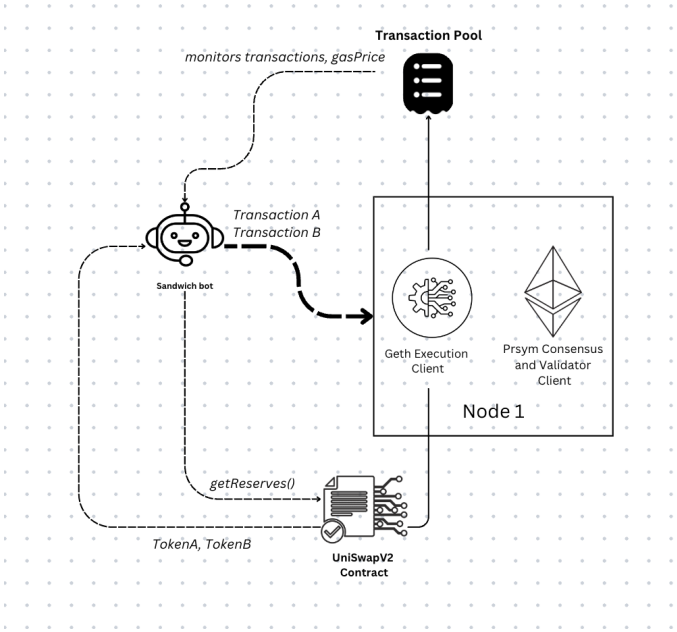


FIGURE 4.2. "Sandwich Bot Operation: The sandwich bot monitors the transaction pool for swap transactions, then interacts with the UniSwapV2 contract to retrieve token reserves. Once a target transaction is identified, the bot sends two strategically priced transactions to 'sandwich' the victim transaction"

they enter the pool, a behavior not fully addressed by many other fair-ordering solutions [OrderFairness-ByzantineConsensus][Themis] . By using this bot, we can analyze the effectiveness of the P3 protocol in countering these types of adversaries and mitigating their impact on transaction ordering and market manipulation in decentralized finance (DeFi) environments.

The sandwich bot operates by continuously monitoring the transaction pool for specific types of transactions, primarily high-value swaps on decentralized exchanges. When the bot detects a suitable target transaction, it retrieves token reserves from the UniSwapV2 contract to estimate slippage and calculate potential profit from manipulating the transaction's price impact.

Attack Execution Process: The sandwich bot follows a systematic approach to executing a sandwich attack, as illustrated in Figure 4.2:

- *Monitoring and Identification:* The bot listens for incoming transactions in the pool, filtering for high-value swaps or trades that meet predefined criteria. It evaluates transaction attributes such as gas price and trade value to identify the most profitable targets for sandwiching.

- *Transaction A (Front-Running):* Upon identifying a target, the bot submits a front-running transaction that is strategically priced and positioned to execute immediately before the victim transaction in the block. This initial transaction manipulates the token reserves in the UniSwapV2 pool, artificially increasing the slippage for the target transaction.
- *Target Transaction Execution:* Following the bot's front-running transaction, the target transaction (victim's transaction) is executed. Due to the preceding manipulation, the victim's transaction faces unexpected slippage, often resulting in a less favorable trade execution price.
- *Transaction B (Back-Running):* After the target transaction executes, the bot submits a back-running transaction to rebalance its position, capturing the profit generated from the price differential. This transaction returns the token reserves to their approximate original state, completing the sandwich attack.

A key part of the bot's operation involves interacting with the UniSwapV2 contract to gather real-time reserve information. By calling the `getReserves()` function, the bot retrieves current token reserves for the assets involved in the trade. This data enables the bot to calculate the optimal transaction amounts for both the front-running and back-running transactions, maximizing its profit from the sandwich attack.

4.2 Protocol Integration and Block Permutation Implementation

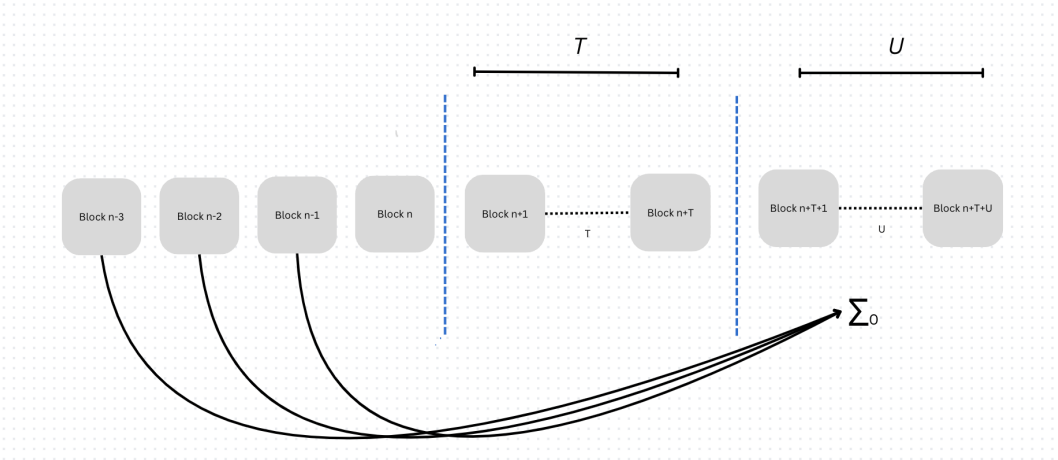


FIGURE 4.3. Generation of the ordering: Partial seeds from leaders of the preceding block are combined to generate pseudorandom value for block ordering

4.2.1 Commitment Tracking and Seed Generation Contract

To facilitate the pseudo-random ordering of transactions without altering the consensus protocol directly, we implemented a smart contract, `MinerCommitmentTracker`, which serves as a decentralized mechanism for tracking commitments and generating seeds in our test network. This contract emulates the functionality required by the P3 protocol by tracking commitments from recent miners and using these commitments to derive the final seed for transaction ordering. The contract enables the pseudo-random transaction permutation while bypassing the need for consensus modifications.

4.2.1.1 Purpose and Approach

The P3 protocol requires that miners commit partial seeds in advance for blocks they may influence in the future, storing these commitments on-chain to ensure transparency and prevent tampering. Under the original protocol, these commitments would need to be recorded prior to block mining, with block validity dependent on miners' compliance. However, implementing this directly on Ethereum would necessitate changes to the consensus protocol or post-mining verification mechanisms—neither of which are feasible within a standard Ethereum test environment.

To address this, the `MinerCommitmentTracker` contract simulates commitment functionality by storing partial seeds from the last three miners in each block. This setup allows the Geth execution client to access these seeds at runtime, leveraging them to derive a pseudo-random ordering for transactions. Although this approach diverges from the Byzantine fault-tolerant aspect of P3, it meets our experimental needs for evaluating non-adversarial behavior and MEV mitigation.

4.2.1.2 Contract Structure and Functionality

The `MinerCommitmentTracker` contract consists of three primary components:

(1) **Commitment Registration:** Each miner records their commitments using a function which accepts an array of commitments (partial seeds) from the miner, stores them, and discards the oldest commitment in the array to maintain a sliding window of the last three miners. This structure mimics the leader set in P3, ensuring that recent commitments are always available for seed derivation.

(2) **Commitment Opening:** The `open` function enables miners to reveal their stored commitments. Although P3 would typically use a deterministic period for opening commitments

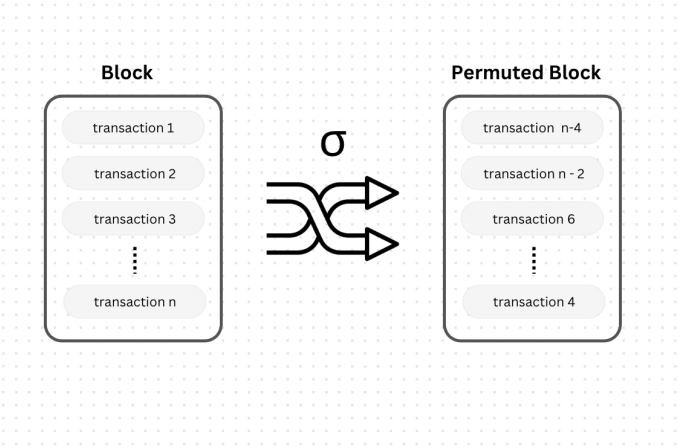


FIGURE 4.4. Permuted Block Ordering: The initial block sequence is rearranged using a PRG function based on XOR-combined values from leader nodes

(silent and loud phases), our contract allows miners to open commitments on demand, recording each opening with metadata such as block number and miner address. This functionality is essential for observing the commitment history and validating the seeds used in transaction ordering.

(3) **Final Seed Calculation:** To derive the final seed for transaction ordering, the *computeFinalSeed* function hashes the concatenated commitments from the last three miners. Using SHA-256 as the pseudo-random generator (PRG), this function provides the randomness needed for block permutation. The Geth client, accessing this seed at runtime, then applies it to reorder transactions according to the P3 protocol's guidelines.

4.2.1.3 Operational Differences and Justification

Due to limitations in modifying the Ethereum consensus mechanism, our contract does not enforce pre-mining commitments as in P3. Instead, it offers a practical alternative where commitments are stored and accessed within the contract state, and the Geth client utilizes them post-mining to create the desired random ordering. This approach, while not Byzantine fault-tolerant, allows us to evaluate P3's effects on transaction ordering and MEV reduction without compromising network stability or requiring consensus-level changes.

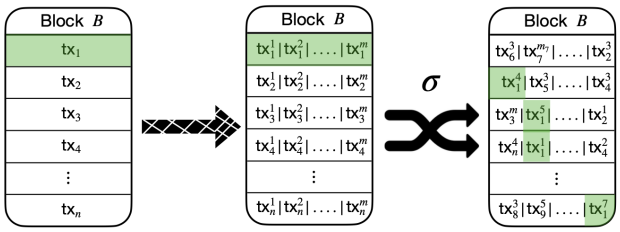


FIGURE 4.5. Illustration of Final Block Permutation Process: Each transaction in Block B is divided into smaller chunks, which are then reordered according to a random permutation (σ)

4.3 Transaction chunking

4.3.1 Implementation of Transaction Chunking

To implement the transaction chunking mechanism required by the P3 protocol, two primary approaches were explored: modifying the Geth client to intercept ERC20 transfer calls and developing a custom router contract to handle swap chunking. Both methods aimed to split larger transactions into smaller, more granular chunks, thereby increasing the range of possible orderings and limiting the effectiveness of potential adversarial manipulations.

4.3.1.1 Intercepting ERC20 Transfers in Geth

The initial approach involved modifying the Geth client to intercept ERC20 transfer transactions at the Ethereum Virtual Machine (EVM) level. This method aimed to track and split each transfer within a swap transaction into smaller chunks, allowing more control over transaction ordering. However, despite detecting and chunking the ERC20 transfers, this approach had significant limitations. The liquidity pool's state changes continued to be calculated based on the original, unchunked transaction data rather than the chunked transfers. As a result, this chunking only appeared on a superficial level, as the underlying state changes did not reflect the smaller transactions. Furthermore, this method required the sender's private key to authorize each chunked transfer, introducing a dependency that would be impractical for real-world usage.

4.3.1.2 Custom Router Contract for Swap Chunking

To address the limitations of the ERC20-based approach, a custom router contract was developed. This contract receives swap function calls, divides the swap amount into smaller chunks, and sequentially executes each chunk on behalf of the user. By handling the swap process within the contract itself, this approach eliminates the need for private key access to authorize individual chunks and ensures that the state changes in the liquidity pools accurately reflect the chunked transactions, effectively implementing the chunking mechanism required by the protocol.

While this approach worked effectively in the test network setup, it is worth noting a potential caveat: in a live network with significant gas costs and high block space demand, the increased number of transactions from chunking could introduce jostling issues. In our controlled test environment, where base gas fees were minimal, this did not present a problem. However, under live network conditions, higher competition for block inclusion could impact transaction costs and prioritization, affecting the practical scalability of this approach.

4.3.1.3 Comparative Analysis and Selection

The custom router contract approach was ultimately chosen due to its ability to manage state changes at the appropriate level of granularity, remove dependency on private key authorization for each chunk, and accurately reflect the impact of chunked transactions on liquidity pool states. This method provided a modular, Ethereum-compatible solution that fulfilled the protocol's requirements and aligned with blockchain's decentralized principles.

CHAPTER 5

Results

Evaluation Results The following table shows the evaluation of the attacker's success and costs during a rushing adversary attack scenario.

5.1 Fatal Front-running

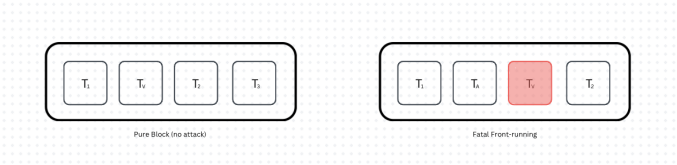


FIGURE 5.1. Victim transaction is front-run by the attacker transaction. As a result, the victim's transaction fails to execute.

To evaluate the effectiveness of a fatal front-running attack, we deployed a smart contract that releases a reward of 0.1 ETH to any user able to correctly submit the pre-image of a given hash function. Upon a user's submission, the transaction becomes visible in the mempool, allowing potential attackers to observe and react before it is mined.

In this experiment, our attacker monitors the mempool for calls to the contract function that reveals the reward, executes the contract locally to verify if the reward is still available, and, if the contract state is still fresh (i.e., open and unclaimed), submits its own transaction with a higher gas price. This higher gas price increases the attacker transaction's priority in Geth, a widely used Ethereum client, which typically selects higher-gas-price transactions to be mined first in a block. By exploiting this mechanism, the attacker consistently places its transaction at the beginning of the block, ensuring it gains priority over the user's transaction.

5.1.1 Vanilla Geth Setup

We conducted 100 trial runs using a vanilla Geth setup, where an attacker with negligible latency (practically zero) front-ran each user transaction. Due to the attacker's priority, every user transaction was outpaced, causing users to lose both the reward and the gas fees paid for transaction inclusion. This sequence of execution arises because each transaction in the block is executed sequentially, causing the global state to change after each execution. Thus, by the time the user's transaction is processed, the contract state has already changed from open to closed, which invalidates the user's attempt to claim the reward. This sequence of events occurs without any way for the user to preempt the attacker's actions, as the user is unaware of the attacker's front-running transaction when submitting theirs.

The metrics from the experiment under the vanilla Geth setup are summarized in Table 5.1. As shown, the attacker success rate was 100%, with the user success rate at 0%. The average profit per successful front-running attack for the attacker was 0.099664 ETH.

TABLE 5.1. Results of Fatal Front-Running Attack on Vanilla Geth Setup

Metric	**Value**
Total Transactions	100
Attacker Success Rate (%)	100.00
User Success Rate (%)	0.00
Average Attacker Profit per Success (ETH)	0.099664

5.1.2 Modified Geth Setup

The same experiment was repeated 25 times on a modified Geth setup. In this configuration, transaction ordering is randomized, which prevents an attacker from consistently prioritizing its transaction over the user's. Importantly, since the transfer of native tokens (in this case, Ether) does not affect the liquidity of a pool, transaction chunking has no impact on the token's value. The only anticipated effect of this modification is the random reordering of transactions, introducing an unbiased, equal probability for the user or attacker transaction to be executed first.

The results of the experiment with the modified Geth setup are presented in Table 5.2. As expected, the success rates were nearly evenly distributed, with the attacker success rate at approximately 48.23% and the user success rate at 51.77%. The standard deviations for both metrics indicate a small variability across trials, reflecting the randomness introduced by the modified setup. The attacker's average profit per successful transaction was reduced to 0.099298 ETH, with minimal deviation, as shown in Table 5.2.

Figure 5.2 provides a visual comparison of the success rates for the attacker and user in the modified setup, including error bars to indicate the observed variability.

TABLE 5.2. Results of Fatal Front-Running Attack on Modified Geth Setup

Metric	**Value**	**Standard Deviation**
Total Transactions	100	N/A
Attacker Success Rate (%)	48.233	4.145
User Success Rate (%)	51.767	3.745
Average Attacker Profit per Success (ETH)	0.099298	0.000065
Average Attacker Gas Cost per Failure (ETH)	0.000336	0

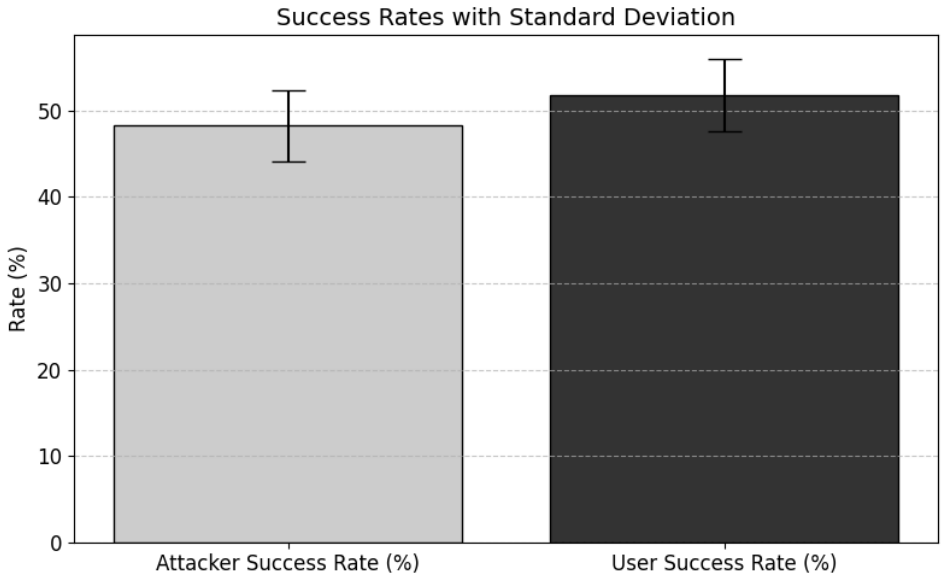


FIGURE 5.2. Comparison of Attacker and User Success Rates on Modified Geth Setup with Error Bars

5.1.3 Analysis of Results

The results from the vanilla and modified Geth setups clearly demonstrate the impact of transaction reordering on the efficacy of front-running attacks. Under the vanilla setup, where transaction priority is determined solely by gas price, the attacker achieves a 100% success rate, completely outpacing user transactions. However, in the modified setup, the introduction of random ordering results in a near-equal success rate for both users and attackers. This reduction in attacker dominance underscores the effectiveness of randomized transaction ordering as a mitigation technique against fatal front-running attacks.

5.1.3.1 Caveats and Limitations

Two key caveats must be considered in the context of these experiments:

1. *Attacker Gas Costs*: In our controlled experiments, we assumed a gas price of 15 Gwei, which aligns with the average gas price on Ethereum under normal network conditions. This gas fee represents the minimum amount the attacker would pay to ensure transaction inclusion in the block. However, in a real-world scenario with multiple bots attempting similar attacks, gas prices would likely escalate as competitors bid for transaction priority. As such, the attacker may need to pay significantly higher gas fees than the user, particularly in a competitive environment. With a roughly 50% success rate in the modified setup, the attacker faces a gamble, as the feasibility of the attack depends on both the contract reward and on-chain gas prices. Although this may reduce the attack's profitability, the user still bears a harder financial loss, as they must also pay gas fees for failed transactions.

2. *Alternative Mitigation Techniques*: While our modified setup introduces random ordering as a mitigation, it is not universally effective. In this specific setup, where transactions are atomic (i.e., the transfer of Ether does not alter the underlying token value), transaction chunking does not apply. However, alternative solutions, such as commit-reveal schemes that utilize encryption, can further secure transactions by obfuscating the pre-image submission phase. Although these solutions fall outside the scope of the present study, they offer additional avenues to mitigate front-running vulnerabilities and will be discussed in subsequent sections.

Overall, these findings illustrate that by merely altering the transaction ordering policy, it is possible to significantly reduce the success rate of rushing adversaries in front-running attacks. The results also suggest that while attackers may still occasionally succeed, the expected profitability of front-running attacks is diminished in a randomized ordering environment. However, real-world applications would need to consider both gas cost dynamics and alternative cryptographic mitigation strategies to provide comprehensive protection against sophisticated attackers.

5.2 Results of Sandwich Attack Simulation

Sandwich attacks involve an adversary monitoring the mempool for transactions, strategically placing transactions before and after a victim's transaction to exploit slippage and extract profit. The mechanics of sandwich attacks were detailed in Section 4. To simulate this attack, we gathered real-world data

from Etherscan, specifically focusing on swap transfers in the WETH/DAI liquidity pool over the last two weeks. The liquidity pool was configured to mirror the actual WETH and DAI reserves for this period, allowing us to accurately model the effects of sandwich attacks under real-world conditions.

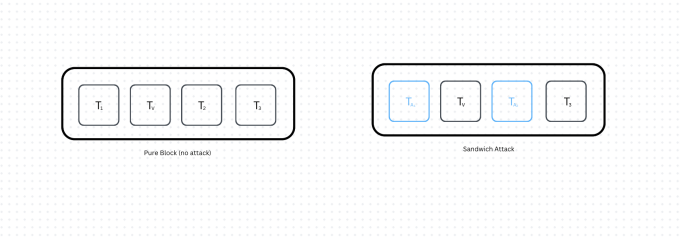


FIGURE 5.3. Victim transaction is front-run by the attacker transaction and also back-run by another attacker transaction. This is the more general case encompassing both tolerant front-running and backrunning

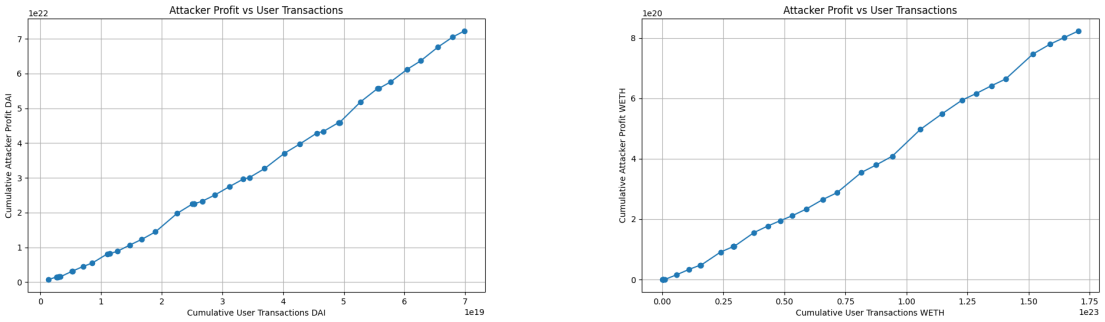
5.2.1 Vanilla Geth Setup with Active Sandwich Bot

We began by running the collected transactions on a vanilla Geth setup with an active sandwich bot. Since the attacker is modeled as a rushing adversary with negligible latency and no competing bots, there is consistently a portion of each user transaction that is "eaten" by the attacker. This results in a guaranteed loss for the user, as the attacker positions its transactions in a way that maximizes slippage exploitation.

It is important to note that the presence of competing bots would not alleviate the victim's losses but would instead create additional losses for the original bot due to the increased competition and rising gas prices. For our experiment, however, we maintained a single bot to isolate the effects of the attack and gather clean data. Figure 6.2 illustrates the relationship between the attacker's profit (equivalently, the user's loss) and the user's transaction amount. When plotted on a cumulative graph, it becomes evident that the primary determinants of the attacker's profit are (a) the amount used by the attacker to sandwich the transaction, and (b) the degree of slippage caused by changes in the reserve ratio. In our dataset, slippage (factor b) is minimal, as no transactions caused substantial changes to the reserves.

5.2.2 Modified Geth Setup with Randomized Ordering and Chunking

To evaluate the effectiveness of our mitigation technique, we ran the same experiment with a modified Geth setup that incorporates randomized transaction ordering and the ability to chunk transactions. The



((A)) Attacker Profits for vanilla Geth (DAI)

((B)) Attacker Profits for vanilla Geth (WETH)

FIGURE 5.4. Comparison of Attacker Profits for Vanilla Geth using DAI and WETH

experiment was conducted in three configurations, each with a different chunk size, to assess the impact of both randomization and chunking on mitigating sandwich attacks.

5.2.2.1 Chunk Size = 1 (No Chunking)

In the first configuration, we used a chunk size of 1, meaning that transactions were not actually chunked. Figure 5.6 displays the results, where significant variability in attacker profit is observed, including swings into negative profit values.

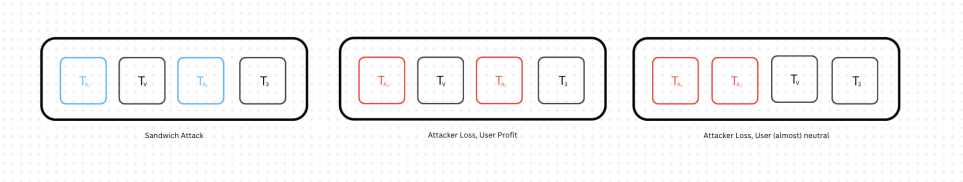
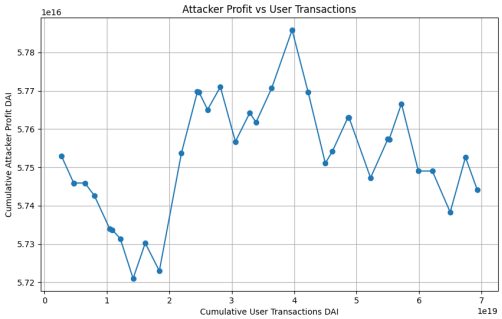


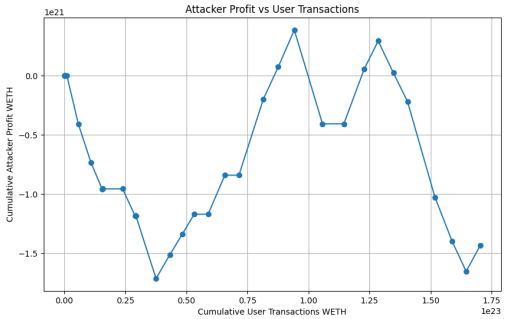
FIGURE 5.5. Possible transaction ordering cases under randomized ordering. (a) Successful sandwiching, (b) reversed ordering (user benefits), (c) neutral ordering with minimal impact.

To understand the occurrences of negative values, we examine Figure 5.5, which shows the three possible scenarios under randomized ordering: 1. *Successful Sandwiching*: This scenario mirrors the vanilla Geth behavior, where the attacker profits by sandwiching the user transaction. 2. *Reversed Ordering*: In this case, the attacker's transactions are placed after the user's, resulting in a loss for the attacker. Effectively, the user gains the value that the attacker would have profited from under normal sandwiching conditions. 3. *Neutral Ordering*: The attacker's transaction backruns the user with an equivalent token amount, producing negligible effect due to minimal slippage and balanced token ratios.

Randomized ordering effectively averages the attacker's profit to near zero, as the probabilities for each scenario are approximately 1/6 for a positive outcome, 1/6 for a negative outcome, and 4/6 for neutral outcomes. Variability in the results can be attributed to randomization and, more importantly, to fluctuations in transaction volumes from the real-world Etherscan dataset.



((A)) Attacker Profits for Modified Geth (DAI)



((B)) Attacker Profits for Modified Geth (WETH)

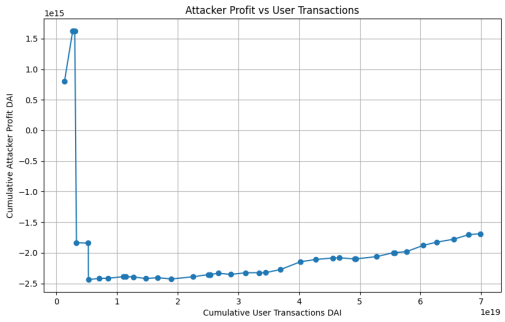
FIGURE 5.6. Comparison of Attacker Profits for Modified Geth using DAI and WETH, with Chunk Size = 1 (No chunking)

5.2.2.2 Chunk Size = 3

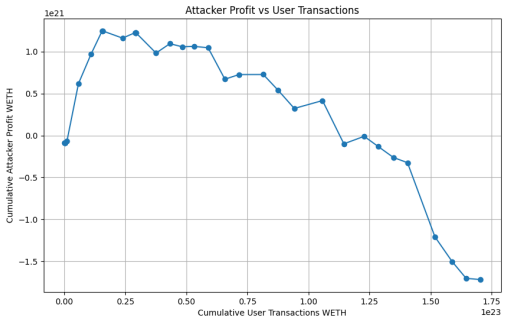
Next, we conducted the experiment with a chunk size of 3, dividing each transaction into three chunks. Figure 5.7 demonstrates that the variability in attacker profit is further reduced compared to the non-chunked setup.

The primary reason for this reduction is the increase in the state space caused by chunking. In the non-chunked setup, a 1/6 probability existed for the attacker to achieve an optimal position relative to the user's transaction. However, with chunking, the number of favorable configurations decreases, making it more challenging for the attacker to consistently benefit from the sandwich attack. Specifically, the probability of an ideal positioning scenario now drops to 1/36, as all three attacker chunks must align advantageously before all three user chunks for maximum profit (as formalized in the original paper). The apparent variations in profit are due to the relative transaction sizes and their cumulative impact on the reserves.

In the final configuration, we increased the chunk size to 6. Figure 5.8 shows that the variability in attacker profit is significantly dampened, resulting in substantially smaller fluctuations compared to the 3-chunk setup.



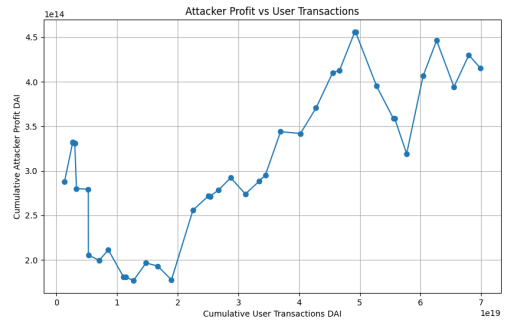
((A)) Attacker Profits for Modified Geth (DAI)



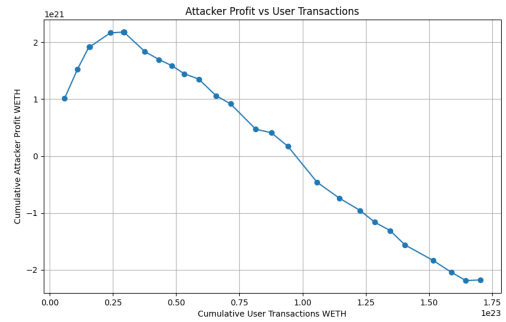
((B)) Attacker Profits for Modified Geth (WETH)

FIGURE 5.7. Comparison of Attacker Profits for Modified Geth using DAI and WETH, with Chunk Size = 3

This configuration represents an extreme case of chunking, where the increased number of chunks further diminishes the probability of an ideal positioning scenario for the attacker. Consequently, the attacker's expected profit per transaction is effectively neutralized. The mathematical principles underlying this outcome follow directly from the probabilities discussed in the 3-chunk setup, with the probabilities further diminishing as chunking is increased.



((A)) Attacker Profits for Modified Geth (DAI)



((B)) Attacker Profits for Modified Geth (WETH)

FIGURE 5.8. Comparison of Attacker Profits for Modified Geth using DAI and WETH, with Chunk Size = 6

5.2.3 Analysis and Observations

The results from the sandwich attack simulation provide insights into the effectiveness of random transaction ordering and chunking as mitigation techniques. In the vanilla Geth setup, the attacker consistently profits by strategically sandwiching the user's transaction, causing predictable and recurring losses for the user. However, in the modified setup, randomization of transaction ordering introduces variability

that disrupts the attacker's ability to maintain a favorable positioning. As a result, the attacker's expected profit becomes averaged out to zero, and for every advantageous position, there are several neutral or unfavorable outcomes.

Furthermore, increasing the chunk size diminishes the probability of a profitable attack, as each chunk must align perfectly for the attacker to achieve the maximum benefit. This exponentially reduces the chances of a successful sandwiching attack, leading to a significant reduction in attacker profit volatility. The impact of chunking becomes more pronounced with larger chunk sizes, as seen in the progressively tamer profit graphs from chunk sizes 1 to 6.

While these findings support the effectiveness of random ordering and chunking in mitigating sandwich attacks, important caveats warrant consideration. While chunking effectively reduces the attacker's profit in atomic transactions, other defenses, such as commit-reveal schemes with encryption, could provide additional protection by obscuring transaction intent and timing, thereby preventing adversaries from precisely targeting victim transactions.

Moreover, there exists one scenario not represented in these graphs; when the sandwich bot is inoperative. Although it may initially appear that this scenario does not influence the efficacy of the mitigation technique, as there is no active adversary to intercept transactions, its implications on the mitigation strategy are non-trivial. As will be discussed in later section, the absence of an active bot introduces considerations that impact the generalizability of the mitigation technique.

CHAPTER 6

Discussion

In this section, we evaluate the effectiveness of the P3 mitigation technique across several critical metrics: Scope, Jostling, Goodput, Decentralization, Delay, and Cost. Additionally, we provide insights into the complexities of transaction chunking, revealing that it is not as straightforward as it may initially seem. Each metric is examined individually to assess the viability of P3 as a defense against various forms of Miner Extractable Value (MEV), such as front-running, back-running, and sandwich attacks.

6.1 Scope

The scope of the P3 mitigation technique evaluates its effectiveness in addressing a range of Miner Extractable Value (MEV) attacks, including front-running, back-running, sandwiching, and other forms of transaction manipulation. Based on our analysis, we rate P3 at 3 out of 4 on a hypothetical scale, where 4 represents a highly robust and comprehensive solution.

6.1.1 Strengths of P3 in Addressing MEV

P3 demonstrates broad applicability in mitigating multiple MEV vectors by introducing randomized transaction ordering and local transaction chunking. These mechanisms significantly reduce the incentive for Priority Gas Auctions (PGA), fostering a more equitable transaction order independent of gas fees. This approach minimizes the effectiveness of most common MEV attacks, including sandwiching and back-running, as confirmed by our experimental results.

Moreover, the ability of P3 to generalize across various Decentralized Finance (DeFi) applications highlights its versatility. By eliminating the reliance on gas fee ranking for transaction inclusion, P3 creates a

more balanced environment for traders, ensuring fairer outcomes without introducing undue complexity or reliance on centralized entities.

6.1.2 Limitations of P3: Fatal Front-Running

While P3 effectively mitigates many forms of MEV, it has notable limitations in addressing fatal front-running attacks. These attacks remain viable because transactions are broadcast to all network participants, allowing a rushing adversary to intercept and replicate the transaction with a higher gas price. The adversary is then able to front-run the user and claim the reward approximately 50% of the time, as demonstrated in our experiments. The expected value of such an attack increases with the reward size, making P3 less practical for defending against high-value, latency-sensitive scenarios.

This limitation highlights a key shortcoming in P3's scope: its inability to obscure transaction content or intent before execution. In this regard, P3 is outperformed by commit-reveal schemes and third-party ordering services like Flashbots or Ethereum 2.0 Proposer-Builder Separation (PBS), which leverage encryption or private ordering to protect transactions from adversarial manipulation.

6.1.3 Comparing P3 to Encrypted Transaction Solutions

Encrypted transaction solutions, such as commit-reveal schemes, generally exhibit a broader scope compared to P3. By concealing transaction details until execution, these techniques effectively neutralize adversarial strategies like front-running and sandwich attacks. However, as discussed in the *Cost* section, this broader scope is primarily a result of the hidden nature of transactions rather than the mechanics of the ordering process itself.

When transactions are hidden, adversaries cannot analyze or manipulate them, which inherently reduces MEV. However, this feature does not address the fundamental challenges associated with transaction ordering in an open, transparent blockchain system. For example, slippage remains an issue in Constant Function Market Makers (CFMMs), even with encrypted transactions. If an attacker blindly places transactions in the absence of visibility, they still face a probabilistic outcome akin to the fatal front-running scenario. This indicates that encryption eliminates some symptoms of MEV but does not resolve the underlying mechanics of transaction ordering.

6.1.4 Conclusion on Scope

While P3 achieves significant progress in mitigating MEV and creating a fairer transaction environment, its limitations in addressing fatal front-running attacks reduce its overall effectiveness. Encrypted transaction solutions offer broader scope but rely heavily on the inherent advantages of transaction obscurity rather than tackling the mechanics of transaction execution and ordering. Therefore, P3 provides a practical, decentralized approach to mitigating MEV but does not yet achieve the comprehensive robustness of encrypted or private transaction solutions.

6.2 Jostling

Jostling measures the level of competition among traders for favorable block positions, particularly in scenarios where transaction inclusion and ordering are influenced by gas price bidding. For P3, we rate its impact on jostling at 4 out of 4, as the technique effectively reduces bidding wars and ensures a fairer allocation of block space. This section explores the factors contributing to P3's performance in minimizing jostling and compares it with other approaches.

6.2.1 Minimizing Gas Price Competition

A key factor contributing to P3's high rating is its local chunking of transactions, which does not increase gas prices or intensify competition for block inclusion. To contextualize this, it is important to understand how Ethereum determines block size. Unlike traditional systems constrained by memory space, Ethereum blocks are limited by gas usage, with each block having a target gas limit of 15 million gas, adjustable up to a maximum of 30 million gas based on network demand. Validators reach consensus on gas limits, with adjustments occurring incrementally by up to 1/1024 per block. The total gas consumed by all transactions in a block must remain within this limit, making gas usage—rather than transaction count—the primary bottleneck in Ethereum's design.

In our experiments, which utilized a common private key and locally chunked transactions, we demonstrated that gas costs could be distributed efficiently across m chunks without exceeding the block gas limit. This finding highlights the scalability of the P3 approach, as chunking does not increase the strain on Ethereum's gas space or transaction throughput, ensuring that P3 can be implemented without disrupting network efficiency.

6.2.2 Comparison with Other Techniques

Compared to private ordering solutions like Flashbots, P3 maintains a decentralized model while achieving similar reductions in jostling. Flashbots and similar systems bypass the public mempool entirely, allowing users to send transactions directly to miners or validators for inclusion in blocks. While effective in preventing gas price competition, these systems introduce centralization concerns and require users to trust third-party services. P3, in contrast, reduces jostling without compromising decentralization, preserving Ethereum's core principles.

Commit-reveal schemes, another alternative, also perform well in minimizing jostling by concealing transaction details until execution. However, these schemes often suffer from increased delays and operational costs, as discussed in the *Delay* and *Cost* sections. P3 avoids these pitfalls by focusing on local chunking and randomized ordering, ensuring a balance between fairness and efficiency.

6.2.3 Conclusion on Jostling

P3 significantly reduces jostling by eliminating the dominance of gas price bidding and introducing decentralized, randomized transaction ordering. Its integration of local chunking and zkProof-based verification ensures that transactions are processed fairly and efficiently, without increasing gas usage or computational overhead. Compared to alternative techniques, P3 achieves a unique balance between minimizing jostling and preserving Ethereum's decentralized design, making it a practical and scalable solution for addressing Miner Extractable Value (MEV).

6.3 Goodput

Goodput, defined as the proportion of genuine transactions processed by the blockchain per unit of time, is a critical metric for evaluating the efficiency of Miner Extractable Value (MEV) mitigation techniques. For P3, we rate goodput at 4 out of 4, reflecting its ability to discourage behaviors that inflate gas prices artificially and prioritize legitimate transactions. By minimizing the impact of Priority Gas Auctions (PGAs) and reducing competition for block space, P3 optimizes block utilization, ensuring that more genuine transactions are included in each block.

6.3.1 Goodput Benefits of P3

A key strength of P3 lies in its local transaction chunking mechanism, which has no negative impact on the number of transactions that can fit within a block. Since chunking occurs locally, it does not introduce additional on-chain storage requirements or computational overhead that might otherwise reduce throughput. This ensures that P3 maintains the overall transaction capacity of the network.

By removing the incentive for adversaries to engage in bidding wars, P3 fosters a more equitable allocation of block space. This reduces the inclusion of attack-related or defensive transactions, such as those used in sandwiching or front-running, allowing for a greater proportion of legitimate user transactions to be processed. As shown in our results, this increase in goodput ensures that P3 enhances network efficiency while preserving its decentralized and fair design.

6.3.2 Comparing P3 to Encrypted Transactions and Other Techniques

Encrypted transaction solutions, such as commit-reveal schemes, also perform well in terms of goodput by eliminating adversarial visibility into the mempool. However, as discussed in the *Cost* section, this advantage arises from the hidden nature of transactions rather than improvements in block utilization mechanics. By concealing transactions, these schemes inherently reduce the likelihood of MEV-related attacks, but they do not address other inefficiencies introduced by adversarial behavior, such as the delays or additional transaction costs associated with failed attempts.

Similarly, while private ordering systems like Flashbots may optimize goodput by bypassing the public mempool entirely, they introduce centralization concerns that conflict with the decentralized ethos of Ethereum. These systems require users to actively seek out private transaction ordering services, which may inadvertently limit access for less experienced users and skew goodput metrics in favor of participants with better resources or knowledge.

In contrast, P3 achieves high goodput while maintaining an open, decentralized blockchain model. By eliminating the dominance of gas fee bidding and introducing randomized ordering, P3 effectively reduces the presence of adversarial transactions without relying on transaction obscurity or centralized third-party services. This makes P3 a more robust and decentralized solution for enhancing goodput.

6.3.3 Caveats: Impact of Chunking on Goodput

While P3 generally maintains goodput, the transaction chunking mechanism introduces a trade-off in certain scenarios. For transactions processed on Constant Function Market Makers (CFMMs), chunking changes the reserve state with each chunk, which can lead to slippage. This behavior does not directly impact goodput but may influence user behavior and transaction patterns. As shown in the *Cost* section, slippage affects both the user and the attacker in equal measure over multiple transactions, resulting in no net gain or loss for the system. However, this dynamic highlights that chunking, while beneficial for fairness, adds complexity to transaction processing in CFMM-based applications.

6.3.4 Conclusion on Goodput

P3 achieves high goodput by minimizing the inclusion of adversarial transactions and preserving block throughput through local chunking. Unlike encrypted transaction solutions or private ordering systems, P3 does not rely on obscuring transaction details or bypassing the mempool, ensuring a fairer and more transparent transaction process. While chunking introduces minor complexities in CFMM scenarios, its overall impact on goodput is positive, making P3 a practical and efficient solution for improving blockchain transaction throughput without compromising decentralization.

6.4 Delay

Delay, which we define as the latency between a transaction's submission and its execution, presents a significant drawback for the P3 mitigation technique. Although not explicitly quantified in our study, P3 introduces additional latency due to the time frame required before blocks are finalized. This delay can be particularly problematic in blockchain networks that require rapid asset transfers, such as Decentralized Exchanges (DEXs). In this section, we evaluate the impact of delay by comparing P3 to other mitigation techniques, with a focus on those that excel in minimizing latency.

6.4.1 Techniques That Excel in Delay

As highlighted in previous studies, protocols that rely on commit-reveal schemes often suffer from increased delays between transaction submission and execution. For instance, delays of even a few blocks can lead to significant price discrepancies. These delays hinder the ability of these protocols

to accurately reflect real-time market prices, resulting in potential arbitrage opportunities and market inefficiencies. While on-chain commit-reveal schemes provide broad scope, their associated delays introduce additional costs, primarily through failed transactions caused by volatility. For example, DEX transactions with slippage tolerance enforced by users may fail if market conditions change significantly during the delay.

An alternative class of techniques excels in minimizing delays while achieving comparable outcomes in cost and scope. One such approach employs (l, n) threshold signature schemes. In these schemes, a single public key encrypts transactions, and l out of n committee members must combine their signatures to decrypt the messages. (Son et al., 2024). Users encrypt their transactions with the public key, and the committee orders the encrypted transactions. Assuming a two-thirds majority of the committee is honest, the members collectively decrypt the messages after finalizing the order.

These threshold-based techniques do not rely on delays in finalization to achieve fairness. Consequently, they avoid the failed transaction issue inherent to on-chain commit-reveal schemes. However, the requirement for compensating the ordering committee introduces an operational cost. While the delayed reward mechanism used in P3 ensures unbiased results through delayed finality, applying a similar approach to threshold schemes without finality introduces challenges in ensuring fairness and impartiality.

Another approach that minimizes delay is the extended UTXO (eUTXO) model. Unlike Ethereum's account-based model, the eUTXO model stores a user's coins in individual unspent transaction outputs (UTXOs) linked to their account. Transactions take UTXOs as inputs, destroy them, and output new UTXOs. For a DEX, the liquidity pool's liquidity is stored in an eUTXO, simulating a smart contract. When a user interacts with the pool's eUTXO, a new eUTXO is created, and the old one is destroyed. This design permits only one interaction with the pool per block, effectively preventing front-running by ensuring no transaction executes on a new state. While the eUTXO model excels in mitigating delay-related issues, it performs poorly on other metrics such as throughput and general applicability. For instance, the limited transaction throughput in eUTXO-based DEXes makes it unsuitable for high-frequency trading or diverse DeFi applications. Therefore, while eUTXO represents an interesting alternative, its narrow scope limits its practicality as a generalized mitigation technique.

6.4.2 Delays Introduced by P3

The delays introduced by the P3 mitigation technique are integral to its design and intended functionality, as discussed in Section 4. Specifically, P3 introduces a delay of $T + U$ blocks, where T represents the number of leaders in the leader set, and U is an additional block required for combining commitments into a seed for a pseudorandom generator (PRG). In a real Ethereum blockchain, where each block takes approximately 12 seconds, a leader set of just 5 nodes would result in a delay of 60 seconds. For time-sensitive applications, such as Decentralized Exchanges (DEXs), this level of latency is unacceptable, necessitating a reevaluation of the randomness generation mechanism used in P3.

6.4.2.1 Challenges with Current Randomness Generation

The primary source of delay in P3 arises from the sequential commitment and reveal phases of the leader set. While this design ensures fairness in generating the random seed, it also introduces significant overhead. Developing a custom randomness generation scheme that minimizes this overhead requires sophisticated mathematical constructs. Fortunately, prior research into verifiable randomness generation offers viable alternatives.

Ethereum's move from Proof-of-Work (PoW) to Proof-of-Stake (PoS) introduced the RANDAO scheme for randomness generation. RANDAO, based on the Commit-Reveal Scheme (CRS), combines commitments from validators to produce a random number. However, as pointed out in (**?**), RANDAO suffers from the Last Revealer Attack (LRA), where attackers with sufficient stake (e.g., 36% of the total staked ETH) can bias the random output to manipulate block proposers. This vulnerability undermines the integrity of the randomness generation process, making RANDAO unsuitable for robust implementations of P3.

To address these issues, Verifiable Delay Functions (VDFs) have been proposed as a solution. VDFs prevent validators from determining the final random number before the reveal phase, thereby mitigating vulnerabilities like LRA. Variants of VDFs, including minimal VDF (mVDF) confirmed by the Ethereum Foundation, offer secure randomness but require specialized hardware known as 'Rig.' While this hardware introduces trust concerns and potential centralization risks, VDFs represent a significant improvement over RANDAO.

An alternative to VDFs is Shamir's Secret Sharing (SSS) algorithm (Shamir, 1979), which securely distributes random seeds among participants. By integrating SSS-based RANDAO into P3, it is possible

to achieve a controllable security level while maintaining decentralized properties. These advances demonstrate that it is feasible to generate secure entropy for randomness without excessive delays.

6.4.2.2 Proposed Modification to P3

To make P3 practical for real-world deployment, we propose modifying its randomness generation mechanism to reduce the delay to a single block. Specifically, we suggest replacing the multi-block commitment-reveal phase with a RANDAO-based mechanism, leveraging either VDF or SSS enhancements for secure randomness. Under this revised scheme, the randomness seed for the next block is generated in the current block, eliminating the need for sequential leader commitments.

With this adjustment, the delay introduced by P3 would be reduced from $T + U$ blocks to a single block, corresponding to approximately 12 seconds in the Ethereum blockchain. This makes the delay comparable to the latency of vanilla Ethereum transactions, addressing one of the primary concerns with P3's feasibility for time-sensitive applications.

6.4.2.3 Comparison with Other Techniques

By reducing the delay to a single block, P3 would fare significantly better than techniques like time-locked puzzles (e.g., those proposed by Doweck and Eyal (Doweck and Eyal, 2020)). Time-locked puzzles utilize Verifiable Delay Functions (VDFs) to achieve fairness but often introduce unbounded delays due to synchronization issues and network latency. Unlike these approaches, the proposed P3 modification ensures a deterministic upper bound on delay, maintaining usability for high-frequency trading and other latency-sensitive DeFi applications.

Moreover, the proposed modification retains the core advantages of P3, including decentralized randomness generation and fairness in transaction ordering. By aligning its delay with standard Ethereum block times, P3 becomes a more practical solution for mitigating Miner Extractable Value (MEV) while addressing the latency issues associated with its original design.

6.4.3 Conclusion

Delays in transaction finality remain a critical challenge for the P3 mitigation technique. However, by integrating proven mechanisms such as VDFs or SSS-enhanced RANDAO, P3 can achieve secure randomness with significantly reduced latency. The proposed modification lowers the delay to a single

block, making P3 more suitable for real-world networks while preserving its core functionalities. This adjustment represents a significant step toward making P3 a viable and efficient solution for mitigating MEV across diverse blockchain applications.

6.5 Decentralization

Decentralization measures the extent to which a mitigation technique impacts the blockchain's decentralized nature. In the context of Ethereum, decentralization ensures that no single party has undue influence over transaction ordering or block creation, with miners (or validators in Proof-of-Stake) acting as independent entities responsible for block construction.

6.5.1 Impact of Centralized Mitigation Solutions

Several existing mitigation approaches to MEV rely on centralized mechanisms, which reduce decentralization by design. These include trusted third-party transaction ordering services such as Flashbots (Daian et al., 2022), Eden (Network, 2024), and OpenMEV (OpenMEV, 2022), which bypass the public mempool entirely. In these systems, users send transactions directly to the third-party service, which is tasked with ordering them. These ordered bundles are then sent directly to miners or validators for block inclusion. While this protects users from front-running attacks by removing their transactions from the public mempool, it requires users to trust both the ordering service and the miners or validators who include these bundles in the blockchain.

The Ethereum 2.0 Proposer-Builder Separation (PBS) model also adopts a similar private ordering approach. In PBS, users send transactions to a block proposer rather than broadcasting them to the public mempool. The proposer assembles ordered transaction bundles and submits them to the miner or validator. While this prevents front-running if the proposer is honest, it introduces reliance on a centralized entity for transaction ordering. The most centralized are solutions that rely on trusted execution environments (TEEs), such as AION (Al-Bassam et al., 2020), tesseract and fairy (Kelkar et al., 2019; Stathakopoulou et al., 2021). These approaches use secure hardware to enforce fair ordering and resist front-running by processing transactions in isolated environments. However, they depend on the integrity of the hardware manufacturer and the assumption that the TEE cannot be compromised, introducing significant trust requirements.

6.5.2 P3's Decentralization Impact

While P3 avoids many of the pitfalls associated with centralized solutions, there are valid concerns about its reliance on the leader set for generating the randomized ordering of transactions. At first glance, this reliance may appear to compromise decentralization, as the system depends on the honesty of the leader set to prevent collusion or bias in the permutation process.

However, P3 incorporates mechanisms to enforce decentralization and deter collusion within the leader set. The key decentralizing feature is the requirement that commitments from previous leaders are only revealed after the miner has committed to a block ordering. Furthermore, block rewards are withheld until the block is added to the blockchain, ensuring compliance with the protocol. To prevent collusion, any node in the leader set whose commitments can be opened by another node before the designated reveal time is removed from the leader set. This mechanism ensures that no leader can prematurely disclose their commitment without risking exclusion from the system, thereby maintaining fairness and decentralization.

The decentralization achieved by P3 also extends to transaction ordering, as the randomization ensures that no single entity can dictate the sequence of transactions. By introducing randomness and delaying the reveal of commitments until after the block ordering is finalized, P3 minimizes the influence of individual leaders or miners

6.5.3 Trade-offs Between Decentralization and Effectiveness

While centralized approaches like Flashbots and PBS offer robust protections against MEV, they significantly reduce the decentralization of the blockchain. These systems shift trust from the decentralized Ethereum network to third-party ordering services or hardware manufacturers, creating potential points of failure. Conversely, P3's decentralized design ensures that transactions remain accessible to all network participants, promoting a fairer and more equitable transaction process.

Nonetheless, P3's effectiveness in certain scenarios, such as fatal front-running attacks, is limited compared to centralized solutions. Trusted ordering services excel in these cases by completely removing transactions from the public mempool, a feature that P3 cannot replicate without compromising decentralization. Similarly, solutions like TEEs offer strong guarantees against front-running but at the cost of requiring trust in proprietary hardware or algorithmic committees, which end up up.

6.6 Cost

Cost measures the additional expenses incurred by traders when executing transactions under the P3 mitigation technique. These costs could stem from increased on-chain storage requirements or separate fees paid to those in charge of ordering transactions. Understanding the cost implications of P3 requires analyzing its effects on various aspects of transaction execution, such as storage, computational overhead, and transaction chunking. This section provides an evaluation of these overheads and discusses their implications in the context of mitigating Miner Extractable Value (MEV).

6.6.1 Overheads in P3

The overheads associated with P3 primarily arise from the commitment and reveal phases of the protocol. In terms of storage, each block in P3 contains nl commitments and, on average, nl openings of partial seeds. A commitment to a partial seed requires 256 bits of storage. Assuming that openings are implemented as smart contract calls in the form $\texttt{open}(i, j, \sigma_{i,j})$, where i and j are 16-bit integers and the address is 160 bits, each opening consumes 468 bits. As a result, P3 incurs an average overhead of $724nl$ bits per block. For a leader set of $nl = 10$, this translates to an overhead of less than 1 KB per block, which is negligible compared to the overall block size. Additionally, since transaction chunking is performed locally, it adds no space overhead to the block.

From a computational perspective, P3 introduces two main sources of overhead. First, the final permutation of $n_t m$ transactions is computed using `PermFromRandBits()`, which has a linear-logarithmic bit complexity of $O(n_t m \cdot \log(n_t m))$. Second, $n_t m - n_t$ additional transactions are executed, incurring an overhead of $O(n_t m)$. Together, these operations scale as $O(m \log m)$, where m is a parameter of P3, making the computational requirements manageable within the existing infrastructure.

The delayed reward mechanism further ensures that leaders participate in the randomness generation process without requiring additional incentives. As a result, there is no additional cost for incentivizing leaders, which simplifies the implementation while maintaining fairness.

6.6.2 Impact of Transaction Chunking

Transaction chunking introduces a unique cost aspect related to slippage, particularly in cases where no other transactions are included in the same block. When a transaction is chunked into smaller parts,

each chunk operates on a new reserve state, leading to slippage. This slippage affects the execution price of each chunk, potentially resulting in either a loss or gain depending on the direction of the swap. Importantly, this behavior is intrinsic to Constant Function Market Makers (CFMMs) and cannot be entirely mitigated by randomization or chunking alone.

Figure 6.1 illustrates the cause of slippage in chunked transactions. Instead of a single large transaction, multiple smaller transactions are executed sequentially, each adjusting the reserve state before the next transaction is processed. This sequential execution creates opportunities for slippage, which is a fundamental characteristic of CFMMs.

Using data collected from Etherscan, we plot the relative sizes of transactions and their corresponding slippage values (see Figures 6.2(a) and 6.2(b)). As expected, larger transactions experience more significant slippage. The difference in slippage between transactions chunked into three parts versus six parts is negligible, highlighting that increasing the number of chunks beyond a certain threshold does not materially affect slippage.

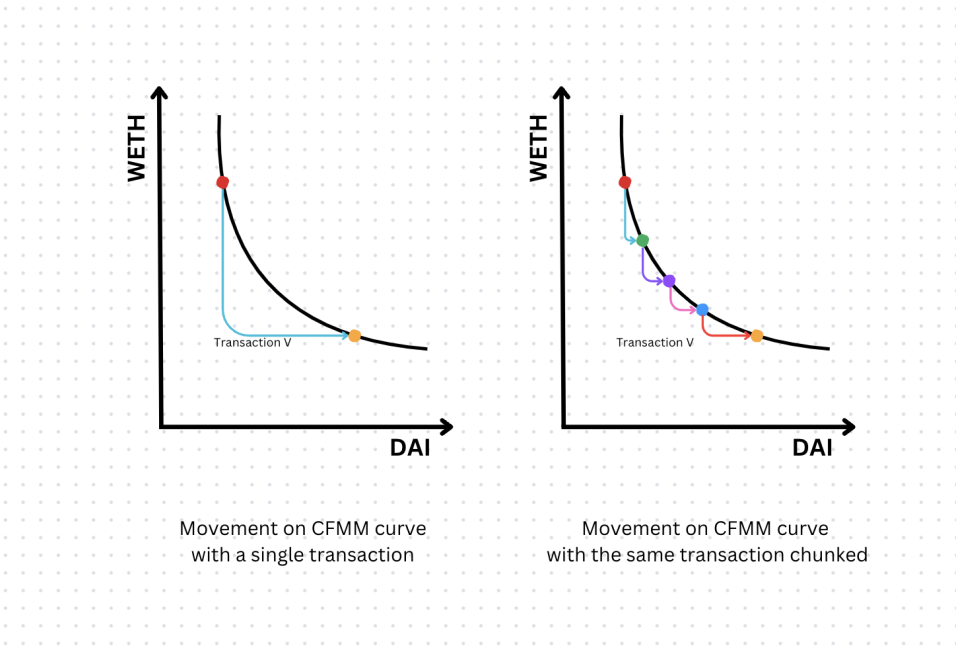
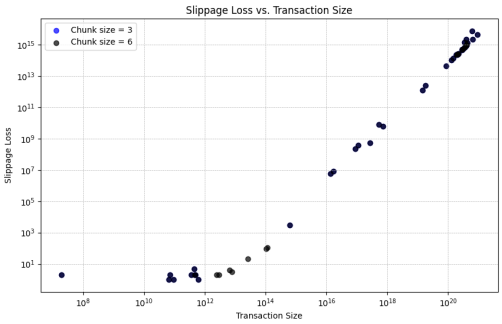
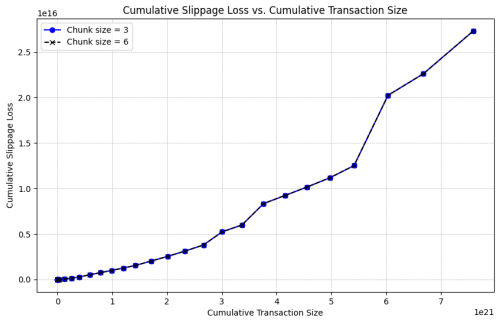


FIGURE 6.1. Illustration of slippage caused by transaction chunking in CFMMs. Each chunk operates on the updated reserve state, resulting in varying execution prices.



((A)) The impact of transaction chunking on slippage loss for chunking sizes 3 and 6



((B)) The impact of transaction chunking on slippage loss for chunking sizes 3 and 6

FIGURE 6.2. Comparison of Attacker Profits for Modified Geth using DAI and WETH, with Chunk Size = 6

6.6.3 Frequency of Transaction Sizes

The slippage impact of chunking is further contextualized by the distribution of transaction sizes. Using the same dataset, we plot a histogram of transaction sizes to analyze their frequency (see Figure 6.3). As expected, most transactions involve swapping small quantities relative to the pool reserves. For these smaller transactions, slippage values are negligible in real-world terms, minimizing their cost impact. This observation underscores that while slippage is an inherent feature of CFMMs, its practical impact is limited for the majority of transactions.

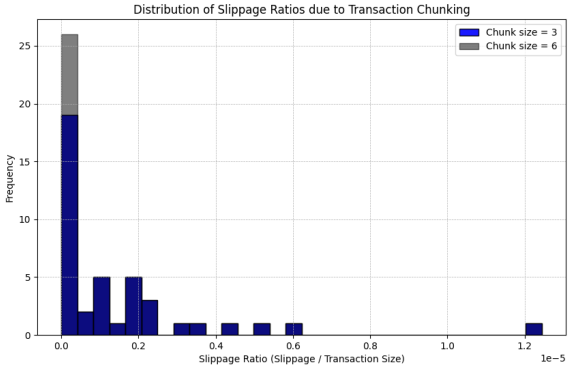


FIGURE 6.3. "The impact of transaction chunking on slippage loss for chunking sizes 3 and 6"

6.6.4 General Observations

The observed slippage is not a flaw of P3 but rather a characteristic of any mitigation technique operating on CFMMs. Even techniques like commit-reveal schemes eliminate the symptoms of sandwich attacks without addressing the underlying cause. For instance, an attacker could still blindly send transactions and achieve either a profit or loss with equal probability, akin to the dynamics of fatal front-running. This highlights that slippage is an unavoidable aspect of CFMM-based DEXes and must be accounted for in any MEV mitigation strategy.

6.6.5 Conclusion on Cost

The cost implications of P3 are minimal in terms of on-chain storage and computational overhead. The delayed reward mechanism effectively incentivizes participation without additional expenses, and the slippage introduced by transaction chunking is an inherent feature of CFMMs rather than a specific limitation of P3. Moreover, the practical impact of slippage is mitigated by the dominance of smaller transactions in real-world trading activity. These findings suggest that P3 achieves a favorable cost profile compared to other MEV mitigation techniques, further solidifying its practicality for widespread adoption.

CHAPTER 7

Threats to Validity

7.1 Internal Validity

The implementation of key components of the P3 mitigation technique was completed, including the transaction ordering and chunking mechanisms. However, the reward mechanism and its implications for leader behavior were not evaluated in detail. Instead, the security analysis of whether the reordering mechanism could be biased or bypassed was taken from the original paper. Additionally, the game-theoretic analysis of Priority Gas Auctions (PGAs) and jostling, which forms a significant part of the argument for P3's effectiveness, was also based on the proofs and findings presented in the original work. This reliance on prior results may introduce confounding factors if the assumptions in the original analysis do not fully align with real-world blockchain dynamics.

7.2 External Validity

The data used for experimentation was drawn from three weeks of transaction activity on a specific pool, the Uniswap v3 DAI/WETH pool. While the dataset was sufficient for the experiments conducted, a broader analysis using data from multiple pools and over a longer time frame could provide a more comprehensive evaluation of the P3 technique's effectiveness. This limitation may affect the applicability of the findings to other decentralized exchanges (DEXs) or less liquid pools.

7.3 Construct Validity

While this research focused on implementing and testing specific components of the P3 technique, it relied heavily on theoretical analyses from the original paper to support claims about its robustness. For example, the proofs and theoretical models used to argue the effectiveness of P3 in mitigating jostling

and PGA may not fully capture nuances introduced by real-world blockchain environments. The lack of an independent security analysis of the reordering mechanism leaves open the possibility of unexamined vulnerabilities.

7.4 Conclusion Validity

The conclusions regarding the efficacy of P3 in addressing MEV, particularly in mitigating jostling and improving goodput, are supported by experimental results and prior theoretical work. However, the reliance on the original paper's proofs and models to justify these conclusions introduces potential biases.

CHAPTER 8

Limitations & Future Work

8.1 Limitations

A primary limitation of the current P3 design is the delay introduced by the multi-block commitment-reveal phase used for randomness generation. This mechanism, while ensuring fairness in transaction ordering, introduces a latency of $T+U$ blocks, where T represents the number of leaders in the leader set and U accounts for additional time required to combine commitments into a seed for the pseudorandom generator (PRG). For example, in an Ethereum blockchain with a block time of 12 seconds, a leader set of 5 nodes would result in a delay of 60 seconds, making P3 impractical for time-sensitive applications such as Decentralized Exchanges (DEXs).

8.2 Proposed Modifications for Future Work

To address the delay introduced by P3, we propose modifying its randomness generation mechanism to reduce the latency to a single block. Specifically, we suggest replacing the multi-block commitment-reveal phase with a RANDAO-based mechanism enhanced by Verifiable Delay Functions (VDFs) or Shamir's Secret Sharing (SSS). This revised scheme would allow the randomness seed for the next block to be generated in the current block, eliminating the need for sequential leader commitments.

Under this adjustment, the delay introduced by P3 would be reduced from $T + U$ blocks to a single block, corresponding to approximately 12 seconds in the Ethereum blockchain. This latency aligns with the delay of vanilla Ethereum transactions, addressing one of the primary concerns with P3's feasibility for time-sensitive applications. By leveraging established randomness generation mechanisms, this approach preserves the fairness and security of P3 while significantly improving its practicality for real-world deployment.

CHAPTER 9

Conclusion

This thesis presented a comprehensive evaluation of the P3 mitigation technique in addressing Miner Extractable Value (MEV) attacks. Through a series of experiments and analyses, we have demonstrated that P3 is highly effective in mitigating most forms of MEV, including front-running, back-running, and sandwich attacks. By implementing local transaction chunking and randomized ordering, P3 reduces the profitability of adversarial strategies and levels the playing field for honest participants.

Despite its effectiveness, P3 is not without limitations. A critical caveat lies in the visibility of transactions in the mempool. This transparency allows adversaries to execute fatal front-running attacks, where transaction chunking is rendered ineffective. In these scenarios, the attacker's probability of success is approximately 50%, as they exploit the visibility to preempt user transactions. While improvements in smart contract security can mitigate some risks, these measures fall short when compared to techniques that obfuscate user transactions entirely. Commit-reveal schemes or encryption-based protocols provide a stronger defense by hiding transaction details until execution.

Future Work

To enhance the practicality and effectiveness of P3, we propose modifications to its randomness generation mechanism. By replacing the multi-block commitment-reveal phase with a RANDAO-based mechanism, augmented with Verifiable Delay Functions (VDF) or Shamir's Secret Sharing (SSS), the delay introduced by P3 can be reduced to a single block. This adjustment would bring P3's delay in line with standard Ethereum transaction latencies, making it more suitable for real-world applications, especially in time-sensitive domains.

Our GitHub repository is made public at "https://github.com/mohammadsaad3o5/ethereum-MEV" with the test setup mentioned above. Any of the experiments described can be recreated using the instructions available.

Bibliography

Mustafa Al-Bassam, Andrea Sonnino, Shehar Bano, Dominik Hrycyszyn, and George Danezis. 2020. Aion: Secure transaction ordering using tees. In *Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 799–815. ACM, London, UK.

Ozan Alpos, Ivan Amores-Sesar, Christian Cachin, and Min Yeo. 2024. Eating sandwiches: Modular and lightweight elimination of transaction reordering attacks. In *27th International Conference on Principles of Distributed Systems (OPODIS 2023)*, volume 286 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Guillermo Angeris, Tarun Chitra, Themis Diamandis, and Kush Kulkarni. 2023. The specter (and spectra) of miner extractable value. *arXiv preprint*, 2310.07865.

Vitalik Buterin. 2023. Ethereum. `https://ethereum.org/en/whitepaper/`.

Andrea Canidio and Vincent Danos. 2024. Commitment against front-running attacks. *Management Science*, 70(7):4429–4440.

Nak Choi and Hyung-Jun Kim. 2024. Decentralized exchange transaction analysis and maximal extractable value attack identification: Focusing on uniswap usdc3. *Electronics*, 13(6):1098.

Philip Daian, Lorenz Breidenbach, and Levy Vecna. 2022. Flashbots: Mev-geth and the democratization of mev. `https://docs.flashbots.net`. [Accessed: Nov. 2024].

Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927. San Francisco, CA, USA.

Yael Doweck and Ittay Eyal. 2020. Multi-party timed commitments: The time capsule protocol. *arXiv preprint arXiv:2005.04883*.

Sheharbano Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. 2020. Sok: Transparent dishonesty: Front-running attacks on blockchain. In Kazue Sako, Steve Schneider, and Peter Ryan, editors, *Financial Cryptography and Data Security*, pages 170–189. Springer, Cham.

Ethereum Foundation. 2024. Ethereum 2.0 proposer-builder separation (pbs). `https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/`. [Accessed: 15-Nov-2024].

Hedera. 2016. Hashgraph: Distributed ledger technology. `https://hedera.com/hashgraph`. [Accessed: 16-Nov-2024].

Mihir Kelkar, Patrick McCorry, Andrew Miller, and Sarah Meiklejohn. 2019. Tesseract: Real-time cryptocurrency exchange using trusted hardware. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1521–1538. ACM, London, UK.

Matthias Keller, Dominik Tschorsch, and Björn Scheuermann. 2019. Themis: Achieving fairness for byzantine fault-tolerant systems. In *Proceedings of the 2019 IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 1233–1242. Dallas, TX, USA.

Klaus Kursawe. 2020. Wendy, the good little fairness widget. *IACR Cryptology ePrint Archive*, 2020(885):1–17. https://eprint.iacr.org/2020/885.

Eden Network. 2024. Eden: The blockchain transaction ordering service. https://v1.edennetwork.io/faq/. [Accessed: 16-Nov-2024].

OpenMEV. 2022. Openmev: Democratizing mev extraction. https://www.openmev.org/. [Accessed: 16-Nov-2024].

Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying blockchain extractable value: How dark is the forest? In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214. San Francisco, CA, USA.

Fabian Schär. 2021. Decentralized finance: On blockchain- and smart contract-based financial markets. *Federal Reserve Bank of St. Louis Review*, 103(2):153–174.

Adi Shamir. 1979. How to share a secret. *Communications of the ACM*, 22(11):612–613.

Dinh Hoai Son, Tran Thi Thao Quynh, and Le Quang Minh. 2024. Randao-based rng: Last revealer attacks in ethereum 2.0 randomness and a potential solution. In *Proceedings of the International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE)*. Hanoi, Vietnam.

Chrysoula Stathakopoulou, Eleftherios Kokoris-Kogias, Lukas Gasser, and Bryan Ford. 2021. Fairy: Fair ordering for consensus protocols using asynchronous trust. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, pages 456–472. IEEE, San Francisco, CA, USA.

Pouyan Zarbafian and Vincent Gramoli. 2023. Lyra: Fast and scalable resilience to reordering attacks in blockchains. In *Proceedings of the 37th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, pages 284–293. https://gramoli.github.io/pubs/IPDPS23-Lyra.pdf.

Yifan Zhang, Eleftherios Kokoris-Kogias, and Ankit Singla. 2020. Byzantine ordered consensus without byzantine oligarchy. In *USENIX OSDI 2020: 14th USENIX Symposium on Operating Systems Design and Implementation*, pages 33–48.

Liyi Zhou, Kaihua Qin, Claudio Ferreira Torres, Duc V. Le, and Arthur Gervais. 2021. High-frequency trading on decentralized on-chain exchanges. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. San Francisco, CA, USA.