Elvis Sikora

An agent-based analysis of commit & reveal schemes to mitigate blockchain extractable value

Brasília, Brasil

2023

Elvis Sikora

An agent-based analysis of commit & reveal schemes to mitigate blockchain extractable value

Dissertação apresentada ao Curso de Mestrado Acadêmico em Economia, Universidade de Brasília, como requisito parcial para a obtenção do título de Mestre em Economia

Universidade de Brasília - UnB

Faculdade de Administração Contabilidade e Economia - FACE

Departamento de Economia - ECO

Programa de Pós-Graduação

Supervisor: Prof. Daniel Oliveira Cajueiro, Dr.

Brasília, Brasil

2023

Elvis Sikora
An agent-based analysis of commit & reveal schemes to mitigate blockchain extractable value/ Elvis Sikora. – Brasília, Brasil, 2023-
45p. : il. (algumas color.) ; 30 cm.

Supervisor: Prof. Daniel Oliveira Cajueiro, Dr.

Dissertação (Mestrado) – Universidade de Brasília - UnB
Faculdade de Administração Contabilidade e Economia - FACE
Departamento de Economia - ECO
Programa de Pós-Graduação, 2023.

1. Inflação de bens industriais. 2. Previsão. 3. Machine learning. II. Universidade de Brasília. III. Faculdade de Administração, Contabilidade e Economia - FACE. IV. Departamento de Economia IV. An agent-based analysis of commit & reveal schemes to mitigate blockchain extractable value

Elvis Sikora

Uma análise baseada em agentes de esquemas commit & reveal para mitigar blockchain extractable value

Dissertação apresentada ao Curso de Mestrado Acadêmico em Economia, Universidade de Brasília, como requisito parcial para a obtenção do título de Mestre em Economia

Trabalho aprovado. Brasília, Brasil, 19 de dezembro de 2023:

Prof. Daniel Oliveira Cajueiro, Dr.
Orientador

Prof. Dr. José Guilherme de Lara Resende
Convidado 1

Dr. Saulo Benchimol Bastos
Convidado 2

Brasília, Brasil
2023

Resumo

A ordem das transações dentro dos blocos em Ethereum afeta os resultados de contratos inteligentes, como decentralized exchanges, oferecendo assim oportunidades para atores mal-intencionados lucrarem às custas dos usuários. Esses ataques contribuem para aumentar os custos, reduzir o rendimento das transações legítimas, e potencialmente ameaçam a segurança da camada de consenso. Por essas razões, inúmeras estratégias têm sido propostas para combatê-los, incluindo mecanismos de confirmação e revelação on-chain. Essas abordagens separam o envio da transação e finalização em blocos distintos enquanto oculta detalhes cruciais da transação durante o envio inicial, tornando os ataques mais difíceis. Expandindo estudos anteriores que investigam antecipação de ordem em finanças descentralizadas com modelos baseados em agentes, esta dissertação visa quantificar o impacto dos atrasos inerentes aos mecanismos de confirmação e revelação na precisão do preço e na potencial perda de lucro para os usuários devido a ataques de reordenação que permanecem viável apesar dessas contramedidas.

Palavras-chave: antecipação de ordens, finanças decentralizadas, modelo baseado em agentes, mecanismo de comprometimento e revelação

Abstract

The order of transactions within blocks in Ethereum affect the results of smart contracts such as decentralized exchanges, thereby providing opportunities for malicious actors to profit at the users' expense. These attacks contribute to increased costs, reduced legitimate transaction throughput, and potentially threaten consensus-layer security. For those reasons, numerous strategies have been proposed to counteract them, including on-chain commit & reveal mechanisms. These approaches separate transaction submission and finalization into distinct blocks while concealing crucial transaction details during the initial submission, making attacks more difficult. Expanding upon previous agent-based studies of frontrunning in decentralized finance, this dissertation aims to quantify the impact of delays inherent in commit & reveal mechanisms on price accuracy and the potential profit loss for traders due to reordering attacks that remain feasible despite these countermeasures.

Keywords: frontrunning, decentralized finance, agent-based model, commit and reveal mechanism

List of Figures

List of Tables

List of abbreviations and acronyms

ABM	agent-based modeling/models.
AMM	automated market maker.
BEV	blockchain extractable value.
CPMM	constant-product market maker.
DeFi	decentralized finance, financial applications run as smart contracts in blockchains.
DEX	decentralized exchange.
LTS	labelled transition system.
TradFi	traditional finance, a term to be used in contrast with the newer, blockchain-enabled, DeFi.

Contents

1 Introduction

Transaction reordering attacks[1] pose significant challenges in decentralized finance (DeFi) applications, constituting a critical consensus-layer security risk and prompting researchers to label it as "a realistic threat to Ethereum today" (Daian et al., 2019). Blockchain extractable value[2] (BEV) represents the potential income block producers (e.g., miners or validators) can acquire by selecting transactions from the mempool[3] and strategically arranging them within a block, sometimes including transactions of their own. Monthly BEV accumulation from decentralized lending platforms and exchanges consistently exceeds $100 million, negatively impacting traders' surplus (Heimbach; Wattenhofer, 2022). This financial burden is further exacerbated by increased gas fees arising from BEV opportunities. Despite considerable research attention, all current mitigation strategies have proven insufficient (Baum et al., 2021; Heimbach; Wattenhofer, 2022; Yang et al., 2022). While frontrunning attacks also affect traditional finance, DeFi faces greater difficulty addressing these issues, as ideally the solutions would be in the protocols, and not depend on regulations.

We investigate limitations that have been pointed out of a specific family of proposed solutions to transaction reordering manipulations, known as on-chain commit & reveal mechanisms (Heimbach; Wattenhofer, 2022). These mechanisms generally operate by separating the submission and finalization of transactions. During the first phase, users commit to their transactions, which are later revealed and finalized by either the user themselves or a smart contract execution. *On-chain* commit & reveal mechanisms, as they name indicate, are fully decentralized, which is essential in DeFi. Additionally, transaction details, such as amounts, remain concealed at the time of submission, seemingly making it challenging to extract any BEV.

However, the delay introduced by commit & reveal mechanisms can significantly impact the accuracy of asset prices in decentralized exchanges (DEXs) (Heimbach; Wattenhofer, 2022). Furthermore, it has been demonstrated that, in chains with public user balances, crafty attackers can still target transactions even when these mechanisms are

[1] The terms "frontrunning" and "transaction reordering attacks" are often used as synonyms in the context of blockchains, which is the convention we adopt. Elsewhere, "frontrunning" is sometimes used to refer specifically to placing an order or transaction ahead of some target transaction. Again, we are interested in the broader family of reordering attacks, which also include, for example, sandwich attacks, which place a swap ahead and another following the target transaction.

[2] Blockchain extractable value is also referred to by other names, most notably "miner extractable value" (MEV) (Baum et al., 2021).

[3] A mempool, or memory pool, is a list of pending transactions that are yet to be added to a block. Each block producer (e.g., a miner, in a Proof-of-Work network) has a mempool. As transactions reach a block producer node, it is expected (but not guaranteed to, due to the decentralized nature of blockchains) to propagate it to the other participant nodes of the network.

in place. With agent-based models (ABMs), we aim to quantify the discrepancy between prices in a constant-product market maker (CPMM) market and an external market for a given pair of assets. We also seek to measure the reduction in traders' surplus when adversaries employ the speculative sandwich attack outlined by Baum et al. (2021). By analyzing these aspects, we aim to contribute to a deeper understanding of the effectiveness of on-chain commit & reveal mechanisms in mitigating transaction reordering attacks and their impact on the DeFi ecosystem.

Our own methodology is built on top of the model from Angeris et al. (2021), who first performed agent-based simulation of DEXs based on CPMMs, which was later adapted to study frontrunning specifically (Struchkov et al., 2021). Struchkov et al. (2021) tried to compare trader's loss to frontrunning attacks from two different DEXs, Uniswap (which uses CPMMs), and Liquifi. Our own work differs from theirs in that we compare CPMMs with and without an on-chain commit & reveal mechanism, and we are interested in price accuracy in addition to trader's loss.

2 Background

In this chapter, we first provide a high-level explanation of BEV to help readers who are not familiar with the topic form a mental model of why it exists at all and how it works. Following that, we move to a review of previous literature on our topic.

2.1 Blockchain extractable value

Blockchain extractable value (BEV) does not exist in a network such as Bitcoin, where everything the miners do is ensure transference of funds between users occurs correctly. Consider a list of transactions of the sort

1. user A is transferring 0.01 BTC to user B,
2. user A is transferring 0.01 BTC to user D,
3. user C is transferring 1.2 BTC to user A.

Now suppose we are to apply these transactions one by one, in the order presented. Each transaction in the list has an effect on users' balances. For example, the first transaction will decrease 0.01 BTC from user A's balance, and add the same amount to user B's. If we shuffle the list, the end result will be the same. Even if we insert some transactions of our own in the middle of the list, the result does not change, other than some of our own funds being transferred that were not before. This, of course, assumes that each user has enough BTC to fund their transfers, but this is guaranteed to be the case if the transactions are to be included in a single block, because were a miner to propose a block where an user overspends her BTC, the other miners in the network would reject that block. Assuming, that is, that enough of the miners are honest and correctly follow the consensus protocol.

Ethereum is not like that. For starters, in Ethereum, users can hold many different types of token, which can represent many different things, as opposed to just ETH. Moreover, there are other types of transaction beyond the simple transfer of funds from one user to another, such as transactions that place computer programs inside the network, which receive the name "smart contracts". In the same manner that miners in the Bitcoin network ensure that transactions are correctly executed, validators in the Ethereum network are responsible for the correct execution of smart contracts. Smart contracts, such as decentralized exchanges (DEXs), can hold their own balances of tokens. Once they live inside the Ethereum network, further user transactions can prompt them, and the software inside them will be executed, possibly moving tokens around. The vision behind DeFi is

to create fully decentralized financial services, relying on modular smart contracts which can interact with one another.

Now, notoriously, Ethereum smart contracts are Turing-complete, meaning that they are powerful enough to execute arbitrary computations. An intuitive way to think of why the order of transactions inside a block matters is to think of what would happen if we shuffled around the lines of a computer program, and inserted new lines in between the existing ones. Unlike the list of Bitcoin transactions presented above, when we shuffle the lines of a computer program, the effect is way harder to predict, and it usually results in a different end state. This example may be too extreme, as reordering lines written in an imperative language will more often than not result in gibberish that errors out (think accessing a variable before it is declared), whereas Ethereum smart contracts are designed to be modular and fault tolerant. So when subsequent invocations to different smart contracts are reordered, the results might differ, but they'll at least be sensible.

Users can trade their tokens using DEXs, which is called "swapping", and when users swap tokens of one type for tokens of another type, the reserves held by the DEX are changed, which affects, in turn, the relative prices of the different tokens. By reordering the transactions that go into blocks, validators may affect the prices different users pay when they interact with the DEX. This open a channel for attacks, such as sandwich attacks, in which a transaction F will be placed just before a target transaction T by an honest user. The goal of F is to affect the prices faced by T such that the honest user pays a worse price than they would otherwise, and the attacker profits from the difference by placing another transaction B following T. This attack arises fundamentally from misaligned incentives, which lead to blocks being built in a way that maximizes profit rather than following the protocols blindly.

2.2 Literature review

DEXs are crucial DeFi platforms that typically utilize automated market makers (AMMs) for their operations. One notable example is the CPMM, which is used by the popular Uniswap exchange (Adams et al., 2021). CPMMs enable users to trade one type of token for another while maintaining a constant product of the two token types' reserves within the AMM, as explained in subsection 3.1.1. Asset prices are not stored explicitly, but, rather, implicitly determined by this rule. Although AMMs are not employed by market makers in traditional financial markets due to their suboptimal nature, they have emerged as the preferred solution for DEXs because of both technical and philosophical constraints imposed by blockchains (Levine, 2022).

DEXs are a cornerstone of decentralized finance (Werner et al., 2021; Qin et al., 2021; Schär, 2021), as they allow users to trade their crypto tokens about as conveniently

as with a centralized exchange, but without needing to entrust their tokens to a centralized entity. Moreover, they work as a credible oracle for prices from the world outside the blockchain. If the prices in external markets do not match those of the AMM, users profiting from the arbitrage opportunity can close the gap. For this reason, smart contracts needing to know those prices can refer to the AMMs. For example, a contract responsible for managing assets on-chain could be programmed to liquidate a position if prices in external markets fall below a certain point. This hypothetical contract could query a DEX to find out about those prices.

DEXs, being central to the DeFi ecosystem, have attracted considerable attention from researchers. Angeris et al. (2021) investigated why CPMMs work as well as they do despite their simplicity. The authors proved several important properties for this class of AMMs, including the fact that manipulating them is costly if liquidity pools are sufficiently large, and that rational arbitrageurs will act to ensure the exchange rate between assets in the AMM accurately reflects market prices. Additionally, they conducted agent-based simulations of market dynamics, demonstrating that many of CPMM's desirable properties are maintained in a more realistic setting than what the theorems assume. Expanding on this work, Struchkov et al. (2021) utilized ABMs to specifically compare the performance of Uniswap and Liquifi, another DEX, under frontrunning attacks. In our own work, we build upon these ABM methodologies to study another proposed family of solutions to decrease value extracted by reordering attacks, commit & reveal mechanisms.

Yang et al. (2022), Baum et al. (2021) and Heimbach and Wattenhofer (2022) wrote recent Systematization of Knowledge (SoK) papers specifically about mitigation strategies for BEV. Various schemes can be categorized under the commit & reveal umbrella, including those with off-chain components. However, we focus exclusively on the on-chain mechanisms, as their off-chain counterparts contradict the principle of decentralization, which seems essential for decentralized finance. Among the on-chain mechanisms alone, numerous proposals have been put forth, and it is unclear which one is superior. Consequently, we concentrate on their shared features. As detailed in chapter 1, we assume that these mechanisms function by separating the submission of transactions from their finalization, ensuring that key transaction details are concealed when first added to the chain. However, in public balance protocols, the user executing a transaction remains visible, allowing adversaries to infer the amount of each token they possess. Additionally, we assume adversaries are still able to control the order of transactions effectively, even without seeing their details.

In chapter 3, we describe the various simulations conducted to test different hypotheses and detail the specific assumptions made about the protocols in each simulation. Heimbach and Wattenhofer (2022) commends the broad applicability and decentralized nature of on-chain commit & reveal mechanisms. However, they criticize these mechanisms

for increasing transaction costs and introducing a delay between submission and settlement, which they argue can cause AMM prices to deviate and result in market inefficiencies. Given the importance of AMMs as price oracles, we focus on measuring the impact of the introduced delay on price accuracy, making an attempt to quantify that impact.

Baum et al. (2021) present an attack that enables adversaries to profit without having access to transaction details, naming it the "speculative sandwich with public user balances". This type of attack involves both frontrunning and backrunning transactions, so it is a "sandwich" attack. The frontrun swap is included in the same block as the user's transaction, while the backrun swap is placed in the subsequent block. The "speculative" aspect of the attack refers to the need for the attacker to guess the direction of the user's swap operation, with public user balances offering clues to inform their guess. The authors demonstrate that this attack is rational for the adversary, but in scenarios with private balances, it would only be profitable if the attacker could accurately guess the direction, which may not always be feasible. In our own study, we implement a different, and suboptimal, speculative sandwich attack, explained in subsection 3.3.4, with the attacker being able to, unlike in Baum et al. (2021)'s, control exactly the order of attacks. We show in chapter 4 that this attack is, in fact, only able to extract a small fraction of the BEV that would be extracted without a commit & reveal mechanism but we caution the reader to interpret this result taking into consideration some of the drawbacks we do not account for in our models, such as the increased transaction costs.

Bartoletti et al. (2021) introduce a conceptually clear and easily implementable formal theory of AMMs in DeFi based on labelled transition systems, a model in theoretical computer science. They prove numerous properties of AMMs, such as the existence of a rational swap that aligns the AMM's internal exchange rate with an external market rate, given certain assumptions. Their model also demonstrates its capability to express BEV attacks. In section 3.1, we review some aspects of this model, while making small modifications to make it more suitable for own purposes since the original version has fixed external market prices, and one of our core aims is to study how well the AMM tracks those prices as they change in different conditions. In a subsequent work building on this foundation, the authors formalize the problem faced by an attacker seeking to maximize their BEV (Bartoletti et al., 2022). Given the AMM state and a pool of pending transactions, the attacker must devise a sequence of transactions, possibly including their own and some from the mempool, to maximize their gains. The authors successfully derive an optimal strategy within the context of CPMMs.

3 Methodology

Our simulation methodology derives from the one first developed by Angeris et al. (2021), which also served as the foundation for Struchkov et al. (2021)'s ABM.

The agents involved are

- liquidity providers, who deposit their tokens into the AMM,
- traders, who use the AMM and perform swaps according to a specific model,
- arbitrageurs, who try to profit from discrepancies between AMM prices and external market prices and
- frontrunners, who monitor pending transactions in the mempool and make transactions of their own to exploit them for personal gain.

There is an Uniswap market, and an external market. The ABM simulation time proceeds as $t = 0, 1, \ldots$. At each time[1] t, an exogenous random price update occurs in the external market, prompting agents to interact with the markets. Transactions with the AMM are not settled immediately, instead, they are added to the mempool.

We seek to answer two questions regarding commit & reveal schemes.

1. How much does the delay introduced by commit & reveal schemes affect the accuracy of AMM prices?
2. How much profit do traders lose to frontrunners executing the speculative sandwich attack with public user balances?

To answer the first question, we conduct two types of simulations. In both, there are no frontrunners and traders have an amount of each token that is large enough so they never run out of tokens. Traders perform their swaps completely randomly. The AMM requires no trading fee.

1. In the first type of simulation, we use the same blockchain model as the one employed by Struchkov et al. (2021). The goal is to measure the accuracy of AMM prices, assuming no delay.
2. In the second type of simulation, we introduce a delay between the submission of transactions and their settlement.

[1] We shall use the terms "time" and "epoch" more or less interchangeably when referring to a period in the simulation when exactly one block is formed and executed.

To answer the second question, we also perform two types of simulations. In both, we utilize the simple blockchain model introduced by Struchkov et al. (2021). The token balances held by traders serve as parameters in the probability distribution from which their trades are drawn.

1. In the first type, we attempt to model a situation without the commit & reveal scheme. Frontrunners can see the details of trades and extract maximum value from them.

2. In the second type, frontrunners cannot view the details of the trades, but only the trader's balances. This allows them to execute speculative sandwich attacks.

3.1 Market model

We adopt Bartoletti et al. (2021)'s model, which we present below. Formally, market semantics are defined using a labelled transition system[2] (LTS). In this LTS, states contain the balances of each token types held by each user, and by the AMM, as well as prices in the external market. Transitions happen due either to transactions involving a user[3] and the AMM, or to nondeterministic price updates. For simplicity, we will refer to price updates as transactions as well. A transition from state Γ to state Γ' after transaction T is denoted $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$. If $\Gamma \xrightarrow{\mathsf{T}} \Gamma'$, we say T is enabled in Γ.

This model of the market as an LTS describes the semantics of all possible ways the markets can evolve, what transactions are enabled in each market state, and what their effects are. The specific trajectories the markets follow will be determined by agent's choices and by random changes in market prices. In the remainder of this section, we first review the basics of Bartoletti et al. (2021)'s theory of AMMs, focusing on the aspects we need. Then, we define the specific probabilistic models we adopt for the price updates.

3.1.1 Market semantics: what transitions are possible

Agents are interested in trading tokens of two types, τ_0 and τ_1, which they can do in two markets, a Uniswap market, and an external market. Besides tokens of these two types, they can also hold tokens of the minted type $\{\tau_0, \tau_1\}$, representing claims on the Uniswap market liquidity pool. For any of the three token types τ, we let $v : \tau$ denote v units of a token of type τ. There is a fixed set of users, an AMM, and the external market. We denote the AMM as $\{r_0 : \tau_0, r_1 : \tau_1\}$ if its reserves are r_0 units of τ_0, and r_1 units of τ_1.

[2] A labelled transition system is a tuple (S, Γ, T), where S is a set of states, Γ is a set of labels, and $T \subseteq S \times \Gamma \times S$ is the labelled transation relation. We write $p \xrightarrow{\alpha} q$ to denote that there is transition from state p to state q with label α, that is, if $(p, \alpha, q) \in T$. (Wikipedia, 2023).

[3] In the context of our market model, we adopt the terms "agent" and "user" interchangeably.

At any specific point during the simulation, the market will be in some state Γ, consisting of

- the balances of τ_0, τ_1 and $\{\tau_0, \tau_1\}$ of each user,
- the reserves $r_0 : \tau_0$ and $r_1 : \tau_1$ of the AMM,
- a price oracle P_Γ, with the external market prices.

Users can swap $x > 0$ units of τ_0 for units of τ_1 (or vice-versa) on the AMM. The amount received back is given by $x\, SX(x, r_0, r_1)$, where SX is the swap rate function, a parameter of Bartoletti et al. (2021)'s model. In our case, since we are modeling a CPMM, we fix

$$SX_\phi(x, r_0, r_1) \;=\; \frac{\phi\, r_1}{r_0 + \phi\, x} \qquad \text{where } \phi \in [0, 1]. \tag{3.1}$$

This way, letting $y = x\, SX_\phi(x, r_0, r_1)$ denote the amount of units of τ_1 the user will receive back from the AMM in exchange for x units of his τ_0 when the AMM reserves are $r_0 : \tau_0$ and $r_1 : \tau_1$,

$$(r_0 + \phi\, x)\,(r_1 - y) = r_0 r_1, \tag{3.2}$$

that is, when $\phi = 1$ and there is no fee, the product between AMM reserves remains constant, and with $\phi < 1$, the user has to give more units of τ_0 than he would without a fee. Uniswap v3 supports multiple fee tiers, including a fee of 0.3%, which was fixed in Uniswap v1, and which was chosen by Angeris et al. (2021) (Adams, 2018; Adams et al., 2021). For simplicity, we fix $\phi = 1$.

The external market prices of τ_0 and τ_1 are, respectively, $P_\Gamma \tau_0$ and $P_\Gamma \tau_1$. Using P_Γ, we can define the external exchange rate X_Γ, which measures how many units of τ_1 one can buy with one unit of τ_0 in the external market:

$$X_\Gamma(\tau_0, \tau_1) \;=\; \frac{P_\Gamma \tau_0}{P_\Gamma \tau_1} \qquad \text{(external)}. \tag{3.3}$$

The internal exchange rate, which we also denote X_Γ[4], is given by

$$X_\Gamma(\tau_0, \tau_1) \;=\; \lim_{x \to 0} SX(x, r_0, r_1) \qquad \text{(internal)} \tag{3.4}$$

if $\{r_0 : \tau_0, r_1 : \tau_1\} \in \Gamma$. In the case of the CPMM we are adopting,

$$X_\Gamma(\tau_0, \tau_1) \;=\; \lim_{x \to 0} SX_\phi(x, r_0, r_1) \;=\; \lim_{x \to 0} \frac{\phi\, r_1}{r_0 + \phi\, x} \;=\; \phi \frac{r_1}{r_0}. \tag{3.5}$$

[4] In Bartoletti et al. (2021), the exchange rate is denoted X, without a reference to the state Γ, and the notation X_Γ is used exclusively for the internal exchange rate. That is because, though they briefly discuss a variant of their model with the price oracle P_Γ depending on Γ, the bulk of their exposition focuses on a version with P being fixed. In the present work, we always use the adjectives "external" and "internal" to distinguish these two concepts.

The supply of token type τ in state Γ is denoted $S_\Gamma \tau$, and defined as the sum of the amounts of τ present in each user's balance and in the AMM. The supply $S_\Gamma\{\tau_0, \tau_1\}$ of the minted token type $\{\tau_0, \tau_1\}$ is used in the definition of the redeem rate $RX^i_\Gamma(\tau_0, \tau_1)$, which governs how many units of τ_i each unit of $\{\tau_0, \tau_1\}$ is worth in redeem transactions, for $i = 0, 1$, and in the definition of the price $P_\Gamma\{\tau_0, \tau_1\}$ of $\{\tau_0, \tau_1\}$. Intuitively, each unit of $\{\tau_0, \tau_1\}$

- gets the user back an amount of the AMM's reserves $r_0 : \tau_0$ and $r_1 : \tau_1$ that is proportional to the supply $S_\Gamma\{\tau_0, \tau_1\}$, in a redeem, and
- is worth an amount that is proportional to the total value of the AMM's reserves.

The redeem rate is defined by

$$RX^i_\Gamma(\tau_0, \tau_1) = \frac{r_i}{S_\Gamma\{\tau_0, \tau_1\}}. \tag{3.6}$$

The redeem rate also governs deposit transactions, as will become clear when we formally define how transactions of each type work. Both redeem and deposit transactions are defined in such a way that the one can be seen as a sort of inverse operation of the other, in a very intuitive way. If a user has just deposited $v_0 : \tau_0$ and $v_1 : \tau_1$, receiving $v : \{\tau_0, \tau_1\}$, they could immediately redeem $v : \{\tau_0, \tau_1\}$, receiving in return $v_0 : \tau_0$ and $v_1 : \tau_1$.

The price of the minted token type $\{\tau_0, \tau_1\}$ in state Γ is given by

$$P_\Gamma\{\tau_0, \tau_1\} \;=\; \frac{r_0\, P_\Gamma\tau_0 + r_1\, P_\Gamma\tau_1}{S_\Gamma\{\tau_0, \tau_1\}} = RX^0_\Gamma(\tau_0, \tau_1) \cdot P_\Gamma\tau_0 + RX^1_\Gamma(\tau_0, \tau_1) \cdot P_\Gamma\tau_1. \tag{3.7}$$

If an user A holds $v_0 : \tau_0$, $v_1 : \tau_1$ and $v_{0,1} : \{\tau_0, \tau_1\}$, with $v_0, v_1, v_{0,1} \geq 0$, we define A's net worth as

$$W_\mathsf{A}(\Gamma) \;=\; v_0\, P_\Gamma\tau_0 + v_1\, P_\Gamma\tau_1 + v_{0,1}\, P_\Gamma\{\tau_0, \tau_1\}. \tag{3.8}$$

If Γ' is a state that is reachable from state Γ, user A's gain between states Γ and Γ' is given by

$$G_\mathsf{A}(\Gamma, \Gamma') \;=\; W_\mathsf{A}(\Gamma') - W_\mathsf{A}(\Gamma). \tag{3.9}$$

We next define all four transaction types in the context of the LTS.

3.1.1.1 Swap transactions

In a swap transaction, denoted $\mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$, user A tranfers $x : \tau_0$, with $x > 0$, to the AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$, receiving in return $y : \tau_1$, which are removed from the AMM, where y is $x\, SX_\phi(x, r_0, r_1)$.

3.1.1.2 Deposit transactions

Deposit transactions are denoted $\mathsf{A} : \mathsf{dep}(v_0 : \tau_0, v_1 : \tau_1)$. User A deposits positive amounts $v_0 : \tau_0$ and $v_1 : \tau_1$ to the AMM $\{r_0 : \tau_0, r_1 : \tau_1\}$, thus becoming a liquidity provider. Unlike swaps, neither deposit or redeem transactions affect the proportion of τ_0 to τ_1 in the AMM. For that reason, in order for a deposit transaction to be permissible, we require

$$\frac{v_1}{v_0} = \frac{r_1}{r_0} \tag{3.10}$$

User A receives back an amount v of freshly-minted token $\{\tau_0, \tau_1\}$ given by

$$v = \frac{v_0}{RX^0_\Gamma(\tau_0, \tau_1)} = \frac{v_1}{RX^1_\Gamma(\tau_0, \tau_1)} \tag{3.11}$$

With this definition, the value $v\, P_\Gamma\{\tau_0, \tau_1\}$ received by A equals the value $v_0\, P_\Gamma \tau_0 + v_1\, P_\Gamma \tau_1$ they provided.

3.1.1.3 Redeem transactions

A redeem is denoted $\mathsf{A} : \mathsf{rdm}(v : \{\tau_0, \tau_1\})$. In a redeem, user A exchanges $v > 0$ of his units of the minted token $\{\tau_0, \tau_1\}$, receiving back $v\, RX^i_\Gamma(\tau_0, \tau_1)$ units of token τ_i, for $i \in \{0, 1\}$. The minted tokens being redeemed are burned. A redeem that would make the supply of minted tokens go down to 0 (or, equivalently, which would deplete the AMM's reserves) is illegal. Similarly to deposit transactions, in a redeem, the value received back by A is exactly equal to the value they are providing.

3.1.1.4 Price update transactions

A price update is denoted $\mathsf{price}(p_0, p_1)$. If Γ is a state and $\Gamma \xrightarrow{\mathsf{price}(p_0,p_1)} \Gamma'$, then Γ' is equal to Γ except that $P_{\Gamma'}\tau_0 = p_0$ and $P_{\Gamma'}\tau_1 = p_1$. We require $p_0, p_1 > 0$.

3.1.2 Market evolution: how prices change

We adopt the same model for the evolution of market prices as the one from Angeris et al. (2021). In our simulation, we fix τ_0 as the numeraire, whose price is fixed at $p_0 = 1$. Hence, if the price of τ_1 is p_1, the exchange rate becomes

$$X_\Gamma(\tau_0, \tau_1) = \frac{1}{p_1} \tag{3.12}$$

Each time an agent exchanges his tokens in the external market and obtains $v_1 : \tau_1$, with v_1 possibly being negative, the external market price evolves as

$$p_1 \mapsto p_1 + \kappa\, v_1^{1+\xi}, \tag{3.13}$$

with $\kappa, \xi \geq 0$ being parameters of the simulation. Following Struchkov et al. (2021), we assumed external markets were large compared to the agents' transactions, letting

$\kappa = 0, |\xi| < \infty$, so that the price update described in equation 3.13 does not happen. Additionally, after each time step, once the agents have performed their actions, prices evolve as

$$p_1 \mapsto p_1 \exp(\sigma X + \mu), \tag{3.14}$$

with $X \sim \mathcal{N}(0, 1)$ being drawn from a standard normal distribution, and μ and σ being the mean returns and volatility of the market when no trades are performed. The agents do not know these rules governing price changes. As in Struchkov et al. (2021), we model both normal market conditions, by setting $\sigma = 0.001$, and critical market conditions, letting $\sigma = 0.03$. We set $\mu = 0$.

3.2 Blockchain model

Because of its simplicity, we base our blockchain model on the one by Struchkov et al. (2021). In their model, new transactions are included in a mempool, which is priority queue sorted by transaction price. New blocks are created and incorporated into the chain when there are enough transactions in the mempool. The required number we used was 100, which is less than what is typical in Ethereum[5], but which is more tractable. For the simulation with delayed transactions, we add a delay between the time when the block is created, and the time when it is considered finalized. During that delay, market prices can evolve, and agents can fire more transactions.

There is a delay parameter. With a delay value of n, when the mempool gets full, the current block is formed and then added at the end of a queue which holds exactly n blocks. The oldest block is then dequeued, and its transactions are executed. The most crucial comparisons are between the no delay condition, which would correspond to $n = 0$, and a 1-block delay, since the latter is what is usually proposed in commit & reveal mechanisms. Despite that, we investigate the effects of higher values for the delay parameter as a way of measuring how much more of a disconnect arises between the markets when more time passes between the moment agents commit to their transactions and the time when it is actually executed. Those results can be seen as a finer grained measure of the sensitivity of results as market volatility varies than just the stark division into normal and critical markets. This is because with a higher value of delay, more market updates happen before a transaction issued by an agent is finalized. Higher values of delay can also be seen as a way of simulating higher block size limits, since more transactions are issued before a given one gets to become finalized.

For the simulations with attackers, they can always inspect the mempool to see who are the agents who issued a transaction, and see the user's balances of each token type.

[5] The exact number of transactions in an Ethereum block is not fixed, but it is usually on the order of thousands.

They also know the probabilistic distribution the agents use to choose what transactions to make. Furthermore, in the simulation with frontrunners but no commit & reveal, the attackers are assumed to also be able to see the details of the transactions.

3.3 Agent types

In this section, we explain the roles performed by the agents populating our simulations.

3.3.1 Liquidity providers

Angeris et al. (2021) describe two liquidity provider models, *initial* liquidity providers, and *rational* liquidity providers. Initial liquidity providers deposit τ_0 and τ_1 into the AMM just before the simulation starts, and then do nothing except hold their minted tokens $\{\tau_0, \tau_1\}$. Rational liquidity optimize a portfolio over the three token types by solving the problem

$$\begin{aligned} \max_{x \in \mathbb{R}^3} \quad & \hat{\mu}^T x - \frac{\lambda}{2} x^T \hat{\Sigma}\, x \\ \text{subject to} \quad & x_1 + x_2 + x_3 = 1 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \tag{3.15}$$

where

- $\hat{\mu} \in \mathbb{R}^3$ is the vector of exponentially-weighted returns for each of the three token types, with respect to the reference market,
- $\hat{\Sigma} \in \mathbb{R}^{3\times 3}$ is the exponentially-weighted covariance of returns and
- $\lambda > 0$ is the penalty incurred by the risk.

Struchkov et al. (2021) only have initial liquidity providers in theirs, arguing that their aims are to study short-term effects, and that liquidity providers typically will want to keep their assets locked-in for a while to accumulate enough fees. That same rationale seems to justify our choice to follow that decision and also include only initial liquidity providers.

3.3.2 Traders

As in Angeris et al. (2021), the trader represents demand for liquidity. We deem it appropriate to, as did both the aforementioned authors and Struchkov et al. (2021), employ *zero-intelligence* traders. That is because we do not care very much about comparing trading strategies. Instead, we want to measure how the proposed protocol affects price

slippage and the distribution of profits. So our trader agents will draw an amount from a probability distribution, and perform a swap in the Uniswap market. Their random trades can go in both directions (i.e., swap τ_0 for τ_1, or vice-versa). In some of our simulations, as previously explained, we include traders whose random transactions depend on their balances. Those can also be considered zero-intelligence, since their trades do not depend on the current market situation, but, to distinguish them from the type we just discussed, we call them *directional* traders.

Our zero-intelligence traders use a Bernoulli distribution with $p = 0.5$ to decide which token type to send into the AMM, and then draw from an uniform distribution to determine the amount being sent, with the minimum and the maximum being some fixed fractions of their total balance. When they issue their swap, they determine the amount of tokens they will receive from the AMM, and set some fixed fraction of that amount as the minimum they require for the swap to succeed. If when the time the swap is executed the state has changed so that they would receive less than that, that is, the slippage exceeds their slippage tolerance, the transaction fails. Directional traders differ only in that the parameter they use for the Bernoulli distribution used to determine swap direction is $p = v_1/(v_0 + v_1)$, where v_i is their balance of token type τ_i. This way, if they have relatively more τ_1 than τ_0, they'll be more likely to send units of τ_1 to the AMM, receiving back some amount of τ_0.

3.3.3 Arbitrageurs

Both Bartoletti et al. (2021) and Angeris et al. (2021) prove optimal arbitrage results. We adopt the one from Bartoletti et al. (2021) because it is slightly more straightforward given that we are already thinking of our markets using their theory. In the case of exchanging τ_0 for τ_1, arbitrageur A will perform a swap $\mathsf{A} : \mathsf{swap}(v, \tau_0, \tau_1)$ if and only if

$$v = \sqrt{\frac{P_\Gamma \tau_1}{P_\Gamma \tau_0} r_0 r_1} - r_0 \tag{3.16}$$

In their simulation, Angeris et al. (2021) add a penalty for large trades to the objective of the arbitrageurs, intending to model risk aversion. For now, we intend to stick to risk-neutral arbitrageurs, as did Struchkov et al. (2021).

3.3.4 Frontrunners

We include in our simulation without commit & reveal Struchkov et al. (2021)'s *insertion front runners*. They estimate the slippage of trading transactions, and then, knowing their slippage tolerance, devise an optimal attack. They front run the trade by placing a transaction that allows it to still succeed, but at the worst possible price. And

then they sell back the assets at a better price. The formulas used to create the attack are given in appendix A.

We call the attacker in the simulation with commit & reveal the *speculative attacker*. They know the trader's balances and the distribution the trader uses to choose transactions, and use that to compute the expected amounts for the swaps the trader would make in each of the two possible directions. If he is confident enough in the direction of the swap, which will happen if the trader holds disproportionally tokens of one of the two atomic types, he will compute the optimal attack targetting the expected swap in that direction, and place his front- and backswap in the mempool, carefully setting their priorities so they happen precisely before and after the user's swap.

3.4 Simulation strategy

To generate the results presented in chapter 4, 20 simulations were performed per experiment per condition. In the case of the study of the effects of delay, each simulation had 100 epochs, that is, 100 blocks were fully formed and then executed. The 22 conditions were made by varying the market condition, which could either be normal ($\sigma = 0.001$) or critical ($\sigma = 0.03$), and the delay parameter, which ranged from no delay up to a delay of 10 blocks. The exchange rate snapshots are taken immediately following a random price update, but before the start of the next epoch. This means that there is some inherent noise between the internal and external exchange rates. The expected value of this noise is less than 0.05% even in the critical condition[6].

When it comes to the question of quantifying the effect on BEV of hiding transaction details from the user, there are 4 conditions, corresponding to the two possible market conditions and two attacker types. BEV was calculated by summing the gain attackers had in each attack happening in the simulation. That is, letting Γ be the state before the attacker frontswap and Γ' be the state following the backswap, the gain is given by equation 3.9. Some of the transactions issued by users were not enabled by the time they were executed, which could happen since the state could change in the interim. In those cases, during the execution of a block, we simply ignored those transactions. If a speculative attacker transaction failed, we still let the user transaction be attempted. If the user transaction would fail, we aborted an attempted insertion attack.

For each of the two types of experiment, and for each fixed market condition, the time series of prices from external markets was randomly generated only once and used

[6] Notice that, with $\mu = 0$, $X_\Gamma(\tau_0, \tau_1)$ changes by a factor of $\exp(-\sigma Z)$ following each epoch in the simulation, with $Z \sim n(0, 1)$. Since $-Z$ also has a standard normal distribution, it follows from the formula for the mgf of a normal distribution that the expected change is a factor of $\exp(\sigma^2/2)$. In the more volatile critical condition, with $\sigma = 0.03$, the expected change is, therefore, a factor of less than 1.0005, such that 0.05% is an upper bound on the expected change.

in all simulations. That means that the external exchange rate in a given epoch was the same for all values of delay sharing a single value of σ. This was done so results are more easily comparable across conditions.

One of the metrics we track is the relative difference, in percent, between the internal and external exchange rates, which we define by

$$\frac{|\text{internal} - \text{external}|}{\text{external}} \times 100\%. \tag{3.17}$$

When, for example, the median relative difference for the normal market condition with 0 delay is reported, it is computed by

1. average across each of the 20 simulations done for that condition to obtain a single time series of internal exchange rates, with 100 measurements,
2. compute the relative difference using equation 3.17 for each of the 100 epochs, ending up with a vector of 100 error measurements,
3. and then taking the median value of those 100 measurements.

The reader might object that step 1 would smooth out random errors in the internal exchange rate, making the AMM mechanism, together with the workings of the arbitrageur, look better than it is. We checked a few individual curves at random, and the results were usually qualitatively similar, so we decided to report the aggregated results only, which also have the advantage of not underselling the mechanism in any given condition because of an uncharacteristically bad run.

4 Results

We present first results from the simulations that try to quantify how much more the internal exchange rate diverges from the external one with the adoption of commit & reveal mechanisms. Following that, we show results related to the other question being answered, that is, how much BEV ceases to be extracted when the attacker cannot see transaction details.

4.1 Effects of delay on price accuracy

4.1.1 Normal condition

Figure 1 shows the mean relative difference between the internal and external exchange rates in percent, as a function of delay. Table 1 shows those means, as well as medians, minimums and maximums, for different values of delay, rounded to 2 digits. The error seems to grow more or less linearly, starting at 0.65% and getting up to 3.3%. The most crucial comparison is between no delay and a delay of 1 block, where the mean difference increases by 0.16 and the median by 0.09 percent points. On its face, that difference does not seem that big, though how much loss of accuracy is too much is hard to assess on the abstract given the diversity of applications of blockchains. For example, if an AMM is used as a price oracle by other smart contracts, a small inaccuracy might lead some threshold to be exceeded which leads to big consequences.

As argued in section 3.4, even in the critical condition, if the arbitrage mechanism was working perfectly, we would expect the discrepancy to be on the order of less than 0.05%, so we can safely say that arbitrageurs are leaving money on the table. They are doing *something*, however, as otherwise we could expect prices to diverge even with no delay. That is because, though only failed trades would actually do nothing, the average zero-intelligence trade does not change the AMM's reserves. Therefore, without arbitrageurs, it would be reasonable to expect the internal exchange rate to be, on average, constant and, while the external exchange rate will also, on average, remain constant, there would be no mechanism ensuring that their random fluctuations covary. The fact that not only the mean difference is reasonably low for all values of delay, but that the median is not that different from the mean and, even more so, is lower than the mean up to a delay of 4 blocks, indicates that there is a corrective mechanism at work.

Figure 2 presents the average internal exchange rate for all epochs for each market condition, together with the external market exchange rate. On the left, we see the plot for simulations with no delay, and on the right with delay of 1 block. One sees that the

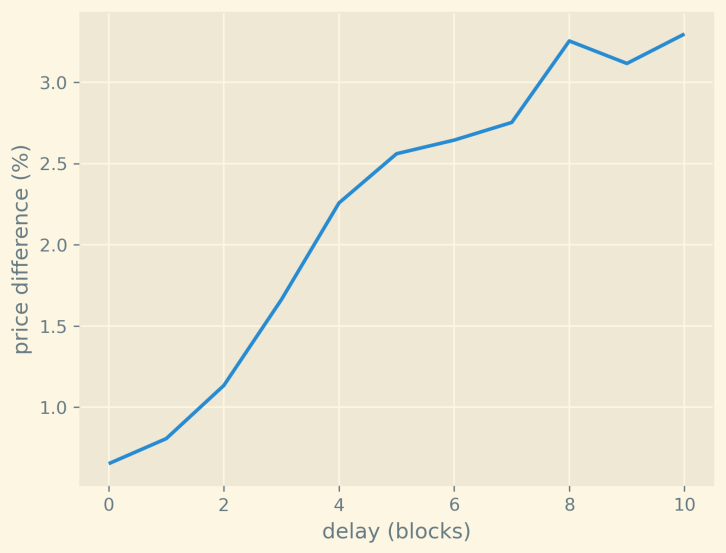

Figure 1 – Relative difference between external and mean internal exchange rates as a function of delay in the normal condition.

delay	median	mean	min	max
0	0.52	0.65	0	2.20
1	0.63	0.81	0	2.80
2	0.97	1.13	0	4.05
3	1.62	1.66	0	4.62
4	2.25	2.26	0	4.84
5	2.95	2.56	0	5.14
6	3.25	2.64	0	5.06
7	3.20	2.75	0	5.05
8	3.42	3.26	0	4.85
9	3.52	3.12	0	4.99
10	3.55	3.30	0	5.35

Table 1 – Statistics describing the relative difference between external and mean internal exchange rates computed using the sample of 20 simulations for each value of delay in the normal condition.

internal exchange rates oscillates around the external exchange rate in both cases, instead of being consistently higher or lower.

In both, the internal exchange rate seems to track its external counterpart, while there is much more dispersion in the simulations with delay up until about epoch 120. That then changes, with the version without delay still displaying oscillatory behavior around the correct values while always remaining at a distance of about 1% or more, while the version with delay seems to often get arbitrarily close to where it should. From epoch 175 on, both seem about equally well-clustered around the external exchange rate. On aggregate, we deem these curves to corroborate the story told by figure 1 and table 1: the internal exchange rate is usually more accurate without delay.

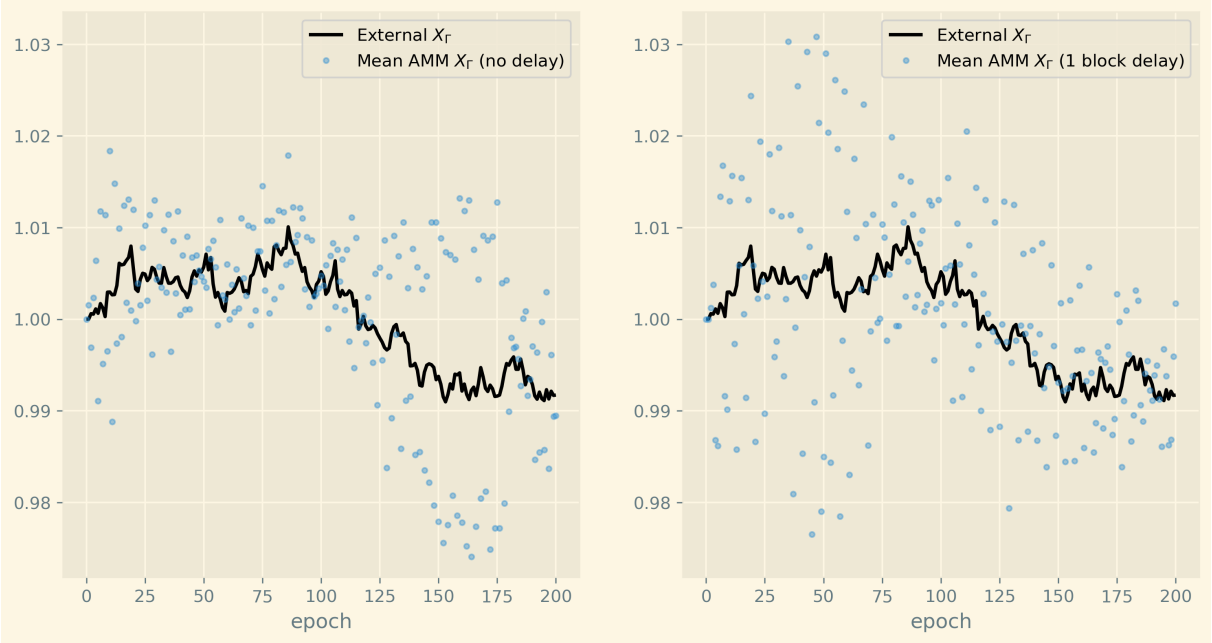

Figure 2 – External and internal exchange rates with no delay (left) and 1 block delay (right) in the normal condition.

4.1.2 Critical condition

Figure 3 is the counterpart in critical market condition to figure 1. The curves for the two market conditions look qualitatively similar, with the one from the the critical condition showing, as one would expect, higher errors for each value of delay. Table 2 shows the same statistics that table 1 did, again, showing an overall similar picture, except with higher errors across the board. The relative difference between the two exchange rates starts already at a mean of 2.75% and a median of 2.34%, which jump starkly to 3.95% and 3.38%, respectively, with a delay of 1 block. This is an unequivocal increase of more than 1%, likely unacceptable for many applications. Even more surprisingly, the maximum relative difference approaches 10% without delay, and exceeds that threshold with it.

Figure 4 shows the internal exchange rate, averaged over the 20 simulations, for each epoch, with and without delay, on top of the curve of external exchange rate, which this time sharply falls to less than half. Again, one sees that the internal exchange rate does track closely the external market prices.

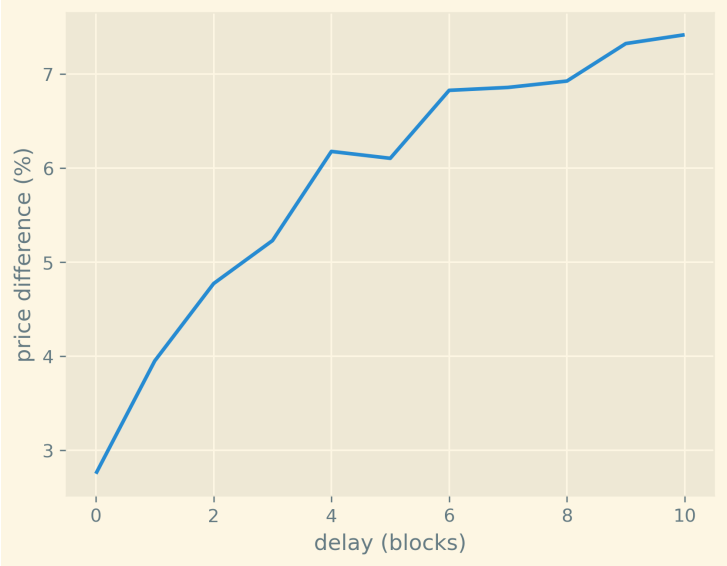

Figure 3 – Relative difference between external and mean internal exchange rates as a function of delay in the critical condition.

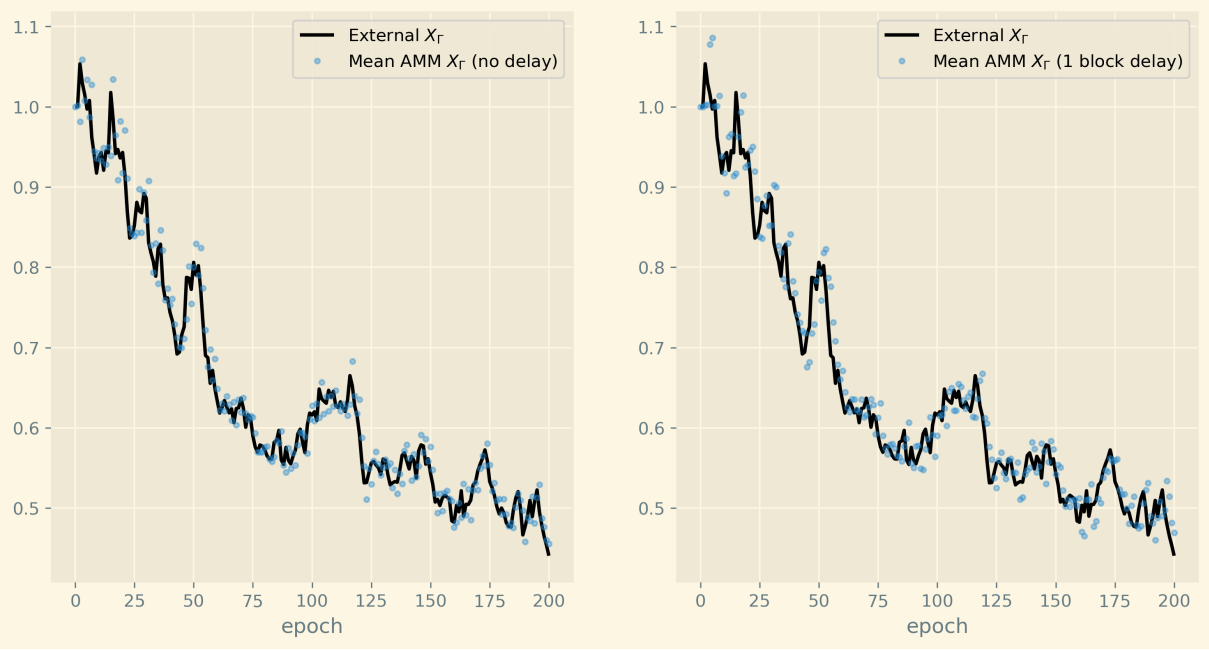

Figure 4 – External and internal exchange rates with no delay (left) and 1 block delay (right) in the critical condition.

delay	**median**	**mean**	**min**	**max**
0	2.34	2.75	0.03	9.28
1	3.38	3.95	0	13.92
2	4.47	4.77	0	16.26
3	4.28	5.23	0	20.12
4	4.95	6.18	0	25.84
5	4.41	6.11	0	25.77
6	5.38	6.83	0	26.07
7	5.19	6.86	0	28.41
8	5.88	6.93	0	30.25
9	6.13	7.32	0	31.52
10	6.62	7.42	0	29.81

Table 2 – Statistics describing the relative difference between external and mean internal exchange rates computed using the sample of 20 simulations for each value of delay in the critical condition.

4.2 BEV difference between speculative and insertion attackers

Table 3 shows the average BEV the different attacker types extracted, averaged across all 20 simulations, in each market condition. The insertion attacker was able to extract 7.1 and 6.6 times as much as the speculative attacker did in the normal and critical market conditions, respectively. This is definitely a big change, which should be weighted against the increase in transaction costs commit & reveal mechanisms cause. While for large trades the higher transaction cost might not make that much of a difference, for smaller ones, it may be a significant composition of the total cost paid by the user making a swap, including BEV. This, together with the loss of price accuracy incurred by those mechanisms, makes cost-benefit analysis of commit & reveal mechanisms non-straightforward.

Interestingly, both types were able to extract more in the critical condition, though neither they or their targets make any reference to external market prices when deciding how to act. The only agent types touching external markets are arbitrageurs, who will change the AMM reserves so they have relatively more of the less valuable token type. Traders will then end up being rewarded with higher amounts of that token type, which attackers can steal, but still, each unit of that token type is less valuable, and it is not obvious that this should lead attackers to profit more. Perhaps the random changes in external prices just happened to create more wealth, which the attackers could gain, in the critical condition than they did in the normal one.

market	attacker	mean	std
normal	speculative	6278.5	1570.6
	insertion	44641.1	782.7
critical	speculative	6911.3	1323.2
	insertion	45408.6	3379.1

Table 3 – Comparing BEV across all conditions.

5 Conclusion

This dissertation has presented an extensive analysis of the impact of commit & reveal mechanisms on internal and external exchange rate divergence and on the reduction of blockchain extractable value (BEV) by attackers. Our simulation results offer valuable insights into the effects of these mechanisms under normal and critical market conditions.

We observed that under normal conditions, the delay introduced by commit & reveal mechanisms causes an increase in the mean relative difference between internal and external exchange rates of 0.16 percent points, starting from a baseline of 0.65%. In critical market conditions, we see a more pronounced jump of 1.20 percent points, starting from an already high baseline of 2.75%, Such inaccuracies, sometimes surpassing the 10% mark, underline the potential risks and challenges commit & reveal mechanisms may pose to market stability and trust, particularly during times of volatility.

These findings are particularly informative for decentralized finance (DeFi) applications, especially those utilizing automated market makers (AMMs) as price oracles, where precision is paramount for maintaining system integrity and preventing cascading effects triggered by threshold breaches. One should notice, however, that good smart contract design should likely be robust to attempted market manipulation.

Our study also highlighted that a lot more BEV gets extracted without commit & reveal mechanisms. The insertion attacker consistently outperformed the speculative attacker, extracting several folds more BEV in both market conditions. This disparity raises important considerations for the design of secure transaction protocols and the balancing of trade-offs between increased transaction costs and reduced vulnerability to attacks.

We conclude that while commit & reveal mechanisms can mitigate certain types of BEV extraction, they introduce trade-offs in terms of transaction latency and price accuracy. The suitability of these mechanisms depends on the specific use case and the relative weighting of transaction cost, latency, and security by the stakeholders. While on their face our results make those mechanisms not look great, at the very least, there might be niches where they could be applied.

For future work, we suggest incorporating a more optimal speculative attack, which would likely increase BEV, while relaxing the assumption that the speculative attacker can guarantee that their transactions will occur in the correct order relative to the user's, which would decrease it. Additionally, an investigation of the effect of varying the number and proportions of agents of each type could be performed. We are unsure about how sensitive our results are to those, as well as to the size of the liquidity pool. It would be

interesting to try and make the internal prices more accurate in the baseline simulation, including no delay and no attackers, only traders and arbitrageurs. Ideally, one could look for inspiration in real-world markets for simulation scenarios. Additionally, while the addition of frictions such as AMM fees and costs for submitting transactions to the AMM might make the proof of the optimal arbitrage swap or optimal attacks more complex, it would make the simulation more realistic, and ABMs are a particularly well-suited vehicle to study their effects.

Bibliography

Adams, H. Uniswap whitepaper. *URL: https://hackmd. io/C-DvwDSfSxuh-Gd4WKE_ig*, 2018. Cited on page 25.

Adams, H. et al. Uniswap v3 core. *Tech. rep., Uniswap, Tech. Rep.*, 2021. Cited 2 times on pages 20 and 25.

Angeris, G. et al. An analysis of uniswap markets. PubPub, 2021. Cited 7 times on pages 18, 21, 23, 25, 27, 29, and 30.

Bartoletti, M.; Chiang, J. H.-y.; Lafuente, A. L. Maximizing extractable value from automated market makers. In: Springer. *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. 2022. p. 3–19. Cited on page 22.

Bartoletti, M.; Chiang, J. H.-y.; Lluch-Lafuente, A. A theory of automated market makers in defi. In: Springer. *Coordination Models and Languages: 23rd IFIP WG 6.1 International Conference, COORDINATION 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021, Valletta, Malta, June 14–18, 2021, Proceedings 23*. 2021. p. 168–187. Cited 4 times on pages 22, 24, 25, and 30.

Baum, C. et al. Sok: Mitigation of front-running in decentralized finance. *Cryptology ePrint Archive*, 2021. Cited 4 times on pages 17, 18, 21, and 22.

Daian, P. et al. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234*, 2019. Cited on page 17.

Heimbach, L.; Wattenhofer, R. Sok: Preventing transaction reordering manipulations in decentralized finance. *arXiv preprint arXiv:2203.11520*, 2022. Cited 2 times on pages 17 and 21.

Levine, M. The crypto story: Where it came from, what it all means, and why it still matters. *Bloomberg Businesweek*, 2022. Available on: <https://www.bloomberg.com/features/2022-the-crypto-story/>. Cited on page 20.

Qin, K. et al. Cefi vs. defi–comparing centralized to decentralized finance. *arXiv preprint arXiv:2106.08157*, 2021. Cited on page 20.

Schär, F. Decentralized finance: On blockchain-and smart contract-based financial markets. *FRB of St. Louis Review*, 2021. Cited on page 20.

Struchkov, I. et al. Agent-based modeling of blockchain decentralized financial protocols. In: IEEE. *2021 29th Conference of Open Innovations Association (FRUCT)*. 2021. p. 337–343. Cited 8 times on pages 18, 21, 23, 24, 27, 28, 29, and 30.

Werner, S. M. et al. Sok: Decentralized finance (defi). *arXiv preprint arXiv:2101.08778*, 2021. Cited on page 20.

Wikipedia. *Transition system — Wikipedia, The Free Encyclopedia*. 2023. <http://en.wikipedia.org/w/index.php?title=Transition%20system&oldid=1147971774>. [Online; accessed 08-April-2023]. Cited on page 24.

Yang, S. et al. Sok: Mev countermeasures: Theory and practice. *arXiv preprint arXiv:2212.05111*, 2022. Cited 2 times on pages 17 and 21.

Appendix

APPENDIX A – Optimal frontswap

Let $\mathsf{T} = \mathsf{A} : \mathsf{swap}(x, \tau_0, \tau_1)$ and Γ be a state such that T is enabled on Γ and $\{r_0 : \tau_0, r_1 : \tau_1\}$ is in Γ. Fix $\phi = 1$ and let $y > 0$ be a number satisfying

$$xSX_\phi(x, r_0, r_1) = x\frac{r_1}{r_0 + x} \geq y \tag{A.1}$$

That is, A receives at least $y : \tau_1$ after performing T.

Let A' be another user and $\mathsf{T}' = \mathsf{A}' : \mathsf{swap}(v, \tau_0, \tau_1)$. We wish to find the value $v > 0$ such that if $\Gamma \xrightarrow{\mathsf{T}} \Gamma' \xrightarrow{\mathsf{T}'} \Gamma''$, then A received exactly $y : \tau_1$ from his swap. Let $\{r_0^i : \tau_0, r_1^i : \tau_1\}$ be the AMM in the intermediate state Γ'. Then, given v, we have

$$r_0^i = r_0 + v, \tag{A.2}$$

$$r_1^i = r_1 - \frac{r_1}{r_0 + v}v \tag{A.3}$$

and we want

$$x\frac{r_1^i}{r_0^i + x} = y. \tag{A.4}$$

Substituting equations A.2 and A.3 into equation A.4, we obtain

$$x\frac{r_1 - \frac{r_1}{r_0+v}v}{r_0 + v + x} = y \tag{A.5}$$

Solving for v results in

$$v = \frac{1}{2}\left(-2r_0 - x \pm \sqrt{\frac{x(4r_0r_1 + xy)}{y}}\right). \tag{A.6}$$

We discard the negative root, as a swap in the direction opposite of the user's could only make them profit more from their swap.