Allocative Inefficiencies in Public Distributed Ledgers [*]

Agostino Capponi[†], Ruizhe Jia[‡], Ye Wang[§]

Abstract

In public blockchains, the visibility of pending transactions can lead to suboptimal blockspace allocation. To mitigate this, users can submit their transactions through private pools to guarantee pre-trade privacy, but they still face execution risk. This study shows that incorporating private pools into public blockchains can reduce allocative inefficiencies and increase overall welfare. However, private pools cannot completely eliminate frontrunning attacks or achieve full efficiency, as validators have strong incentives to maintain rents from frontrunning. To address this, we propose an improvement for the design of private pools that makes it incentive-compatible for all validators to adopt them, eliminate frontrunning, and achieve efficiency.

Keywords: Blockchain; Allocative Efficiency; Market Design; Miner Extractable Value; Private Pool.

[*]We are grateful to Andreas Park (discussant), Chaojun Wang (discussant), Hanna Halaburda, Zhiguo He, and Shihao Yu for valuable comments and insights. We are also grateful to seminar participants at the University of California at Berkeley, the participants of the Chinese University of Hong Kong Distinguished Lecture in Quantitative Finance, and the participants of Tokenomics 2022, and of the 2nd Annual CBER Conference.

[†]Corresponding author: Columbia University, Department of Industrial Engineering and Operations Research, Email: `ac3827@columbia.edu`

[‡]Columbia University, Department of Industrial Engineering and Operations Research, Email: `rj2536@columbia.edu`

[§]ETH Zürich, Department of Information Technology and Electrical Engineering, Email: `wangye@ethz.ch`

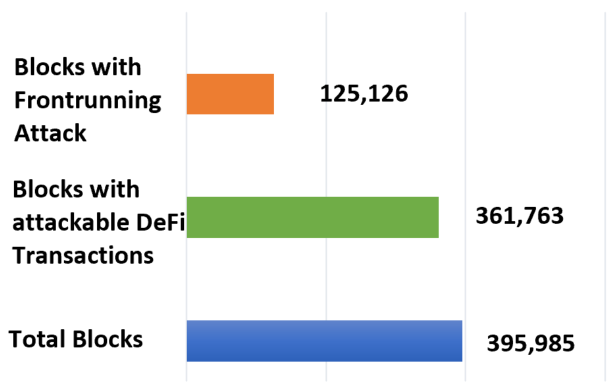
1 Introduction

Blockchain technology has evolved past a platform to support decentralized payment systems (Nakamoto (2008)). Today, it supports a range of services, including tokenization (Cong et al. (2020b)), crowdfunding (Gan et al. (2021)), and decentralized finance (Harvey et al. (2021)). The allocation of transactions to blocks is decided through a fee auction process, which incentivizes validators to prioritize high fee transactions (Buterin (2015), Nakamoto (2008)). However, the visibility of pending transactions before confirmation can compromise the efficiency of such allocation mechanisms.

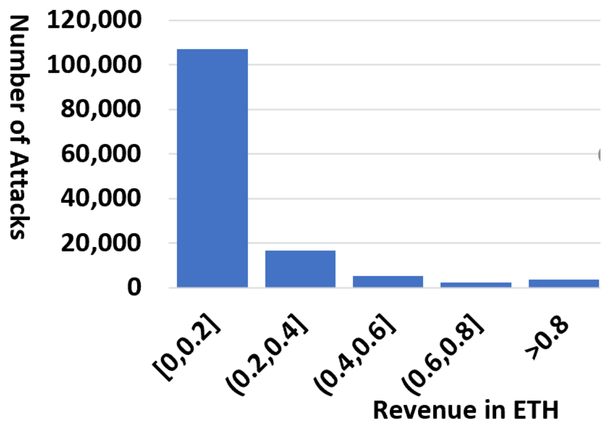
In public blockchains, the transparent observability of pending transactions creates opportunities for frontrunning attacks (Park (2021), Daian et al. (2020)). Malicious actors submit their own transactions with higher fees ahead of the intended victims, "jumping the queue" and prioritizing their executation. We argue that this frontrunning friction leads to a socially sub-optimal allocation of blockspace. Empirical evidence shows that losses due to frontrunning attacks reached over 100 million USD by 12/31/2021 (See figure 1b), making it unsafe for users to submit transactions to a public blockchain. The reduced safety weakens user incentives to submit transactions, and may result in social inefficiency if valuable transactions are excluded from the blockchain. Moreover, frontrunning transactions transfer wealth from victims to attackers but do not generate any economic value. As a result, the allocation of blockspace to fronrunning transactions is a social waste. Empirical estimates indicate that over 30% of existing blocks allocate space to frontrunning transactions (See figure 1a).

Private transaction submission pools have been proposed as a solution to mitigate frontrunning by creating a private channel for directly submitting transactions to validators and making them inaccessible to malicious actors. Implementations of private pools, such as those provided by Flashbots Protects and Eden Network[1], aim to mitigate the negative impacts of frontrunning and maximize welfare (see Daian (2022)). Despite their increasing popularity, it remains uncertain whether validators, who play a critical role in confirming transactions in public blockchains, will have adequate incentives to adopt and accept transactions from private pools. The effectiveness of private pools in mitigating frontrunning and improving welfare is also unclear. This paper investi-

[1]For an overview of Flashbots relay, refer to `https://docs.Flashbots.net/Flashbots-auction/overview/`For an overview of the Eden Network, refer to Piatt et al. (2021).



(a) Distribution of blocks with frontrunning risk and attacks



(b) Distribution of attack profits from frontrunning attacks

Figure 1: Analysis of frontrunning attacks on Ethereum blockchain from 3/1/2021 to 4/15/2021. Panel (a) shows that out of 395,985 total blocks generated, 361,761 blocks contained transactions subject to frontrunning risk, and 125,126 of these blocks were identified as frontrunning attacks. Panel (b) displays the distribution of attack profits, with the average profit being approximately 0.19 ETH or $600 as of 12/31/2021, and total losses reaching over $110 million. The data and methodology used to produce these results are described in the appendix.

gates the welfare implications of incorporating private submission pools into public blockchains and provides insights into how their market design can be improved to enhance allocative efficiency.

Our findings indicate that, in all equilibria, validators adopt private pools to some extent. The incorporation of private pools weakly reduces blockspace misallocation and leads to an increase in aggregate welfare. However, private pools are not able to fully eliminate frontrunning attacks or attain efficiency in situations where they are not fully adopted. This failure stems from validators having a strong incentive to preserve rents generated from frontrunning attacks. To tackle this issue, we offer a proposal to enhance the design of private pools, making it incentive-compatible for all validators to adopt them, eradicate the frontrunning problem, and attain efficiency.

Our model features three types of agents, namely validators, users, and attackers; and two pool types, namely a private and a public mempool. Validators have the option to monitor both pools or just the public one. We consider two groups of users, referred to as *frontrunnable* and *non-frontrunnable users*, who can submit their transactions through either the public or private pool. Users are heterogeneous in their valuations of transactions. The frontrunnable user is subject to frontrunning risk when submitting transactions through the public pool, and her transaction is referred to as the *frontrunnable transaction*. Attackers compete to frontrun her transaction submitted through the public pool.

Transactions in the public pool are visible to all agents, including validators who adopt the private pool, while transactions in the private pool are only visible to its monitoring validators. The likelihood of order execution in the private pool is determined by the rate of validator adoption, while the pool selection of users and attackers affects the benefits derived by validators from monitoring the private pool.

As the number of validators monitoring the private pool increases, the frontrunnable user is more likely to submit transactions to the private pool as the risk of the transaction not being observed by the chain-updating validator, i.e. the execution risk, is outweighed by the advantage of avoiding frontrunning risk.

Validators monitoring the private pool are constrained to prioritize orders submitted through this pool. This intensifies a "race to the front" for priority order execution, and attackers will use both the public and private pools to their advantage. Through the private pool, they gain priority execution, while the public pool ensures guaranteed execution. The competition between attackers drives up fees above the minimum requirement for block inclusion, which are passed on to the validators. The integration of the private pool with the public pool intensifies this competition and increases the rents extracted by validators in each frontrunning attack.

Validators can observe more transactions by monitoring the private pool, but if too many validators join the private pool, the execution risk in private pool becomes low enough to incentivize the frontrunnable user to migrate from the public to the private pool. This migration, in turn, eliminates frontrunning attacks and the associated rents. Thus, it may not be incentive-compatible for all validators to adopt the private pool.

We then analyze the subgame perfect equilibrium of the game. If the frontrunning issue is pronounced, the frontrunnable user would not have any incentive to use the public pool, thereby excluding her transaction from the blockchain in the absence of a private pool option. This results in inefficiency. The unique subgame perfect equilibrium in this scenario is for the user to submit their transaction exclusively through the private pool to avoid frontrunning and for all validators to adopt the private pool in order to monitor the transaction and earn higher fees due to the increased demand for block space. Because a fully adopted private pool makes it safe for frontrunnable users to submit transactions, an efficient block space allocation is achieved with a private pool.

On the other hand, if the frontrunning problem is not significant, multiple equilibria may exist. One such equilibrium involves some validators not adopting the private pool, leading to a high level of execution risk and the frontrunnable user submitting their transaction to the public pool despite the risk of frontrunning. In this case, validators who only observe the public pool would not have sufficient incentives to also observe the private pool, as doing so would eliminate frontrunning and result in lost fees from competing attackers. This results in an inefficient block space allocation, as frontrunning attacks persist, wasting block space.

This adoption failure resembles the classical *hold-up problem*: the benefit of introducing a private pool is entirely appropriated by frontrunnable users in the sequential game, leading to a loss for validators and disincentivizing them to adopt it. To align the incentives of frontrunnable users with those of validators, we propose a contractual solution where the frontrunnable user ex-ante commits to pay an additional fee to validators in the private pool for executing her order. We show that this incentive scheme leads to a unique full adoption efficient equilibrium.

Literature Review. To the best of our knowledge, our paper is the first to explicitly model the allocative inefficiency caused by frontrunning attacks and evaluate the welfare implications of incorporating private submission pools into public blockchains.

Our work contributes to existing literature on friction reduction in public blockchains. There have been several studies on Maximal Extractable Value (MEV) and its mitigation strategies, with frontrunning profits considered a type of MEV that validators and attackers extract at the expense of blockchain users (Auer et al. (2022)). Yang et al. (2022) provides a comprehensive overview of MEV mitigation tools. Park (2021) shows that pricing rules in decentralized exchanges are susceptible to frontrunning attacks in the form of "sandwich" attacks at Automated Market Makers. Lehar and Parlour (2023) examine the competition between "good" and "bad" MEV bots and argue that private settlement can reduce the under-investment problem of "good" MEV bots. Canidio and Danos (2023) and Gans and Holden (2022) propose alternative mechanisms to address frontrunning attacks. Our focus is on the welfare implications of private pools and their impact on the allocative efficiency of blockspace.

Another literature focuses on friction caused by miner collusion and manipulation in the transaction fee market (Chung and Shi (2021); Roughgarden (2021); Lehar and Parlour (2020)) and forks in Proof-of-Work blockchains (Biais et al. (2019)). These studies abstract from the frontrunning problem and only consider public transaction submission channels.

Our paper also relates to the existing literature on economic analysis of transaction fee (Huberman et al. (2021); Easley et al. (2019)), mining strategies (Capponi et al. (2019); Cong et al. (2020a); Prat and Walter (2021)), and consensus protocols (Saleh (2020); John et al. (2020); Roşu and Saleh (2021); Bakos and Halaburda (2021)). While some works focus on block formation mechanisms and security implications, our study focuses on the economics of transaction submission mechanisms. Moreover, our work shows that private pools will change transaction fee bidding strategies and mining strategies as miners gain the option to monitor private pools.

More broadly, in prior work on market design under frictions, Roth (2008) argues that to achieve efficiency, a market needs to provide thickness (Roth et al. (2004)), overcome congestion (Roth and Xing (1997)), and be safe for participants (Niederle et al. (2006)). These principles apply to the design of transaction submission pools for public blockchains, except that agents cannot be restrained from certain behaviors, such as frontrunning, and are not obligated to participate in the new mechanisms, such as private pools. Our paper shows how to improve private pool market design to induce adoption, reduce blockspace waste, and eliminate frontrunning and execution risks, improving allocative efficiency and making blockchain transactions safer for users.

The paper proceeds as follows. We provide the institutional background of private transaction pools in Section 2. We develop the game theoretical model in Section 3. We solve for the subgame perfect equilibrium, and analyze its implications in Section 4. We analyze welfare in Section 5. Section 6 provides empirical supports for our model implications. Proofs of technical results are relegated to the Appendix.

2 Institutional Details of Relay Services

In this section, we delve into the intrinsic information leakage issue in public blockchains and the background information of private transaction pools.

2.1 Blockchain and Information Leakage

A blockchain is a decentralized ledger technology that operates in a peer-to-peer (P2P) network. In this network, agents can issue and broadcast their transactions, which are then collected by validators into blocks. These blocks are added to the existing chain of blocks, forming a chain of records that is maintained by all agents in the network. Transactions in a blockchain network are verified and validated through a consensus mechanism, which ensures that all agents agree on the state of the blockchain.

The cost of executing a transaction on a blockchain network is typically in the form of a transaction fee, which is paid to the validator who includes the transaction in the block. The higher the transaction fee, the more incentivized the validators are to prioritize the transaction in the queue of pending transactions. This results in a transaction fee auction, where agents bid for the opportunity to have their transactions executed first. This mechanism ensures that the limited blockspace on the blockchain is efficiently allocated to transactions that are deemed most valuable by agents Buterin (2015).

One of the key features of blockchain technology is its transparency, which allows all agents in the network to observe pending transactions in their own mempools before they are confirmed and added to the blockchain. However, this transparency can also present a security risk when blockchains support financial services, such as decentralized finance (DeFi) transactions. In these cases, attackers can exploit information about pending transactions to execute frontrunning attacks, which can be costly for agents Eskandari et al. (2019).

Frontrunning attacks can take various forms, including displacement, insertion, and suppression attacks Torres et al. (2021). A displacement attack occurs when an attacker observes a profitable transaction from a victim agent and then broadcasts an identical transaction with a higher fee. This frontrunning transaction will be executed before the victim transaction, allowing the attacker to earn the profit while the victim transaction fails. An insertion attack involves the attacker observing a transaction, such as a token buy order in a decentralized exchange (DeX), from a victim agent. The attacker then broadcasts two transactions: a frontrunning transaction with a higher fee and a backrunning transaction with a lower fee. The frontrunning transaction buys the same token as the victim transaction, resulting in a higher execution price for the victim transaction due to price impact. After the victim transaction is executed, the price of the purchased token goes up again, and the backrunning transaction closes the position by selling the token, resulting in a wealth transfer from the victim agent to the attacker. A suppression attack occurs when an attacker observes a transaction from a victim agent and then broadcasts transactions with higher fees to prevent the victim transaction from being included in the block. This type of frontrunning attack is expensive, as the attacker must use a large amount of blockspace to reach the block capacity limit. Currently, insertion frontrunning attacks are the most common type of frontrunning in the DeFi space Torres et al. (2021).

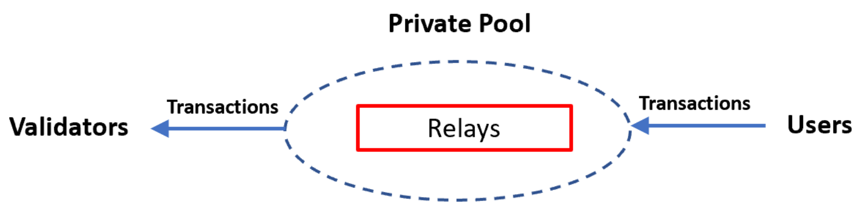
2.2 Relay Services

Relay services represent a form of private transaction submission pool implementation. A centralized relay service receives transactions from users and forwards them to validators, without broadcasting them in the peer-to-peer (P2P) network. This arrangement reduces the risk of malicious actors observing the transactions. Nevertheless, it requires users to trust the centralized relay service, which oversees all transactions that are transmitted through it.

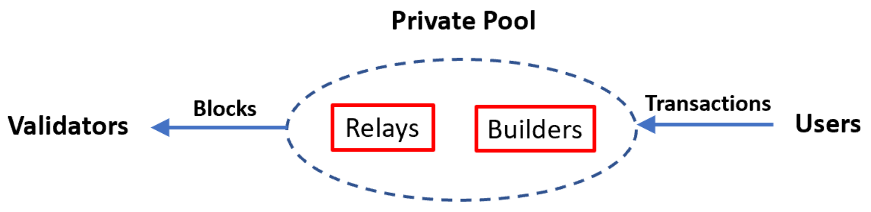
Empirical evidence suggests that this trust requirement is not problematic in practice. Of more than two million transactions submitted through private pools, only 8 have been frontrun by attackers Yang et al. (2022). This validates the frontrunning protection of private pools. The cause of the 8 frontrunning incidents was forking, whereby the block in which the transaction was included was uncled. This phenomenon is referred to as the "uncle bandit risk," and it can be mitigated by carefully verifying parent hashes (see flashbots (2021) for more details).

The first relay service, Flashbots, was established in January 2021. Validators that join Flashbots prioritize the transactions submitted through the Flashbots relay that have the highest bids by including them at the top of the block. The order of execution for transactions is established through a one-round, sealed-bid, first-price auction flashbots (2021). In contrast, transaction fee bidding in public pools follows an ascending price auction format and may involve multiple rounds of bid submissions. Additionally, pending transactions and their associated fees are publicly observable.

After the transition of the Ethereum blockchain from Proof-of-Work (PoW) to Proof-of-Stake (PoS), the implementation of relay services and private pools underwent slight modifications. The new implementation is known as proposer-builder separation (PBS). First, users no longer send their transactions directly to validators, but instead transmit them privately to block builders. Builders collect transactions from the private channel, as well as the public mempool, and pack them into a block, which they then forward to validators. Only validators that have adopted the relay service can receive this block. Second, users are no longer required to trust validators to avoid frontrunning, as validators must sign a blinded block (which only includes block headers) and pass it on to the builder to complete the block content. In this manner, the role of block building is delegated from validators to builders Yang et al. (2022). Figure 2 depicts the different implementations of private pools before and after the merge.



(a) Before Merge Private Pool



(b) Post Merge Private Pool

Figure 2: 2a depicts the implementation of private pools before the merge, and 2b illustrates the implementation of private pools after the merge.

3 Model Setup

The model we analyze consists of three periods, indexed by $t = 1, 2, 3$, and three types of agents: blockchain users, attackers, and validators. All agents are assumed to be risk-neutral.

Validators. There are N rational and homogeneous validators, each with equal probability, $\frac{1}{N}$, of appending the next block to the chain. The validator who appends the next block, referred to as the winning validator, is drawn randomly at the end of period 3. We assume N is sufficiently large. The winning validator earns transaction fees attached to the included transactions.[2] A validator

[2]The fixed reward is not considered in our analysis as it remains constant regardless of the validator's adoption of private pools.

can include at most B transactions in a block due to limited capacity.

There are two transaction submission channels: public and private pools. In period 1, validators can choose to adopt the private pool at no cost.[3] We denote by M and $\alpha = \frac{M}{N}$ the number and proportion of validators who adopt the private pool in period 1. All validators can observe transactions submitted through the public pool, while only those in the private pool can observe transactions submitted through the private pool.

At the end of period 3, the winning validator selects the B transactions with the highest attached fees among those he observes, breaking any ties randomly. If the winning validator has joined the private pool, he is required to prioritize the private pool transactions and execute them in decreasing order of fees.[4] The public pool transactions are then executed in decreasing order of fees. If the winning validator has not joined the private pool, he executes only the public pool transactions in decreasing order of fees.

The validator's adoption decision does not affect his probability of winning the next block, and he decides whether to adopt the private pool to maximize expected transaction fees, conditional on winning the next block. The expected transaction fees earned from adopting the private pool or from monitoring the public pool only are both contingent on the adoption choices of users and attackers. We denote the expected fee revenue of the winning validator by $r_{private}(\cdot)$ if he adopts the private pool, and by $r_{public}(\cdot)$ if he stays on the public pool only.

Users. There are two types of users, whose type depends on the exogenously specified nature of their transactions. The first type of user is frontrunnable, and her transaction is frontrunnable. This user receives a private value v_0 from executing her transaction, but if her transaction is sent through the public pool, then with probability p, attackers recognize the frontrunnable nature of the transaction and front-run it, earning a profit $c \geq 0$ and causing a loss of c to the frontrun user.

The second type of users are referred to as non-frontrunnable users, and their transactions are non-frontrunnable. There are $B + 1$[5] non-frontrunnable users, indexed by $i \in 1, 2, ..., B + 1$, who receive private values v_i from executing their transactions. Without loss of generality, we let

[3]As stated in `https://docs.Flashbots.net/Flashbots-auction/miners/faq/`, the Flashbots relay is an open-source software that does not charge for usage.

[4]This is the requirement imposed by relay services such as Eden and Flashbots.

[5]The results should remain unchanged if there were more than $B + 1$ non-frontrunnable users. If there were less than $B + 1$ of these users, the analysis in many cases would become trivial because there would be no competition for block space.

$v_1 > v_2 > ... > v_{B+1}$. We also let $v_0 > v_{B-1}$ and $(1-p)c > v_{B-1}$ to rule out trivial cases where the frontrunnable user or attackers are unable to overbid other users and get on-chain. To rule out corner cases, we impose the following assumption:

Assumption 1. *The difference $v_{B-1} - v_B$ is sufficiently small.*[6]

In period 2, users decide which pools to submit their transactions to. A user can opt to broadcast her transaction through the public pool, the private pool, or not submit it at all. If a frontrunnable transaction is submitted through the public pool, it is exposed to the risk of being identified and front-run by attackers. On the other hand, if a transaction is submitted only through the private pool, it will not be observed by validators who do not monitor the private pool. The probability of a transaction being included in the next block is at most $\alpha = \frac{M}{N}$, and thus the execution risk is determined by the adoption rate of the private pool among the validators. The frontrunnable user is indexed as user 0, and we denote the pool chosen by user i ($i \in \mathcal{I} = \{0, 1, 2, ..., B+1\}$) by $C_i \in \{\text{Private}, \text{Public}, \text{None}\}$. Each user i attaches a transaction fee f_i to her transaction.

User i selects the submission pool C_i and attached fee f_i to maximize her expected payoff:

$$U_i = \mathbb{E}\left[\mathbb{1}_{\text{Executed,i}}(v_i - f_i) - c\mathbb{1}_{\text{frontrun,i}}\right],$$

where $\mathbb{1}_{\text{Executed,i}}$ is an indicator function for the event that the transaction submitted by user i is included in the block by the validator, and $\mathbb{1}_{\text{frontrun,i}}$ is an indicator function for the event that the transaction submitted by user i is front-run by attackers. In case of a tie, we assume that users will prefer the public pool. This assumption is justified by the fact that submitting transactions through private pools, such as Flashbots Protect, typically requires more sophistication, while the public mempool generally has a more user-friendly interface.

Attackers. In the system, there are two competing attackers, indexed by $j \in \mathcal{J} = \{1, 2\}$. The attackers monitor the public pool for frontrunnable transactions and aim to exploit them, resulting in a profit $c \geq 0$. If a frontrunnable transaction is present in the public pool, both attackers will identify it with probability p.

[6] If there are sufficient transactions, i.e., B is large enough, and the private values that users receive from executing these transactions are drawn from an i.i.d bounded distribution, then the expectation of the difference between the B^{th} order statistic and the $(B-1)^{th}$ order statistic is sufficiently small.

At the beginning of period 3, each attacker decides which pool(s) to submit their order if they have identified a frontrunnable transaction. If an attacker chooses to submit the order to both pools, they will provide the same nonce for both transactions. Since each nonce can only be used once, at most one of the two transactions will be executed.

We denote the pool chosen by attacker j as $V_j \in \{\text{Public}, \text{Private}, \text{Both}\}$. The transaction fee bid by attacker j in the private pool is denoted by f_{D_j}, while the fee bid in the public pool is denoted by f_{L_j}. Attacker j chooses the pool and fee to maximize their expected payoff:

$$A_j = \mathbb{E}\left[\mathbb{1}_{\text{wins,j}} \mathbb{1}_{\text{frontrun,0}}(c - f_{executed,j})\right],$$

where $\mathbb{1}_{\text{wins,j}}$ is the indicator function for the event "attacker j's order is executed before the other attacker's order", and $f_{executed,j}$ is the transaction fee paid by attacker j. The tie-breaking rule for attackers is as follows: "both pools" is their preferred choice, followed by the "public pool", and finally the "private pool".

Transaction Fee Bidding. The order is executed by the attacker who submits the highest transaction fee. The transaction fee bidding mechanisms in the public and private pools differ. In the public pool, commonly referred to as Priority Gas Auctions (PGA), the transaction fee bidding is a variant of an English auction, an open-outcry ascending-price auction. The main difference is that the transaction fee bidding in the public pool ends randomly when the validator decides the new block and stops receiving new transactions.

In period 3, if both attackers submit their orders to the public pool, there is either one round or two rounds of bidding with equal probability, and either attacker could be the first mover with equal probability. In each round, only one attacker moves, and the bid increment must be larger than ϵ. All bids in the public pool are visible to both attackers.

To minimize downside risk, attackers deploy a smart contract, which would terminate the transaction if the attack no longer exists. In such a case, the transaction is deemed as failed and the corresponding fee is negligible and assumed to be equal to zero in our model.

The transaction fee bidding in the private pool is a one-round, sealed-bid, first-price auction, where all bidders only submit their bids once to the relay, without leaking any information to other bidders. If two attackers submit the same order, the transaction submitted by the attacker who attaches the highest fee will be selected by the validators.

Equilibrium. We solve for the subgame perfect equilibrium (SPE) of the game described above. The strategy profile consists of the private pool's adoption decisions by validators, the pool selection and transaction fee bidding strategies of users, and the pool selection and transaction fee bidding strategies of attackers. The strategy of user i is a mapping from the private pool's adoption rate by validators, α, to her pool choice C_i and transaction fee bid f_i. The strategy of attacker j is a mapping from the private pool's adoption rate by validators, α, and users' actions, $(C_i, f_i), i \in \mathcal{I}$, to his selected channel V_j and transaction fees submitted in each pool f_{D_j}, f_{L_j}. A symmetry restriction is imposed such that the strategies used by both attackers are identical.

4 Model Analysis

In this section, we provide a comprehensive analysis of our model, beginning with a discussion of how frontrunning leads to sub-optimal backspace allocation in the absence of a private pool option. We then proceed to solve for the subgame perfect equilibrium (SPE) of the game, taking into account the presence of the private pool. Our analysis includes an examination of the pool selection strategies of both attackers and users, followed by a study of the equilibrium adoption rate of the private pool by validators.

4.1 Allocative Inefficiency without a Private Pool

In this subsection, we investigate the allocative inefficiencies that result from frontrunning attacks in the absence of a private pool. Aggregate welfare is defined as the sum of the expected payoffs of all participants, including validators, users, and attackers. It is important to note that the profits of attackers and fee revenue of validators are transfers of wealth from users, and the transaction fees paid by attackers to validators are merely a portion of the profits that attackers extract from users. As a result, aggregate welfare is equal to the sum of the private values earned by users executing transactions in the blockchain.

An efficient allocation is achieved when blockspace is assigned to users with the highest valuations for their transactions, resulting in aggregate welfare of $\sum_{i=0}^{B} v_i$. The allocation of blockspace in a public pool-only system is hampered by two inefficiencies: transaction exclusion and blockspace waste. Firstly, the elevated frontrunning risk deters frontrunnable users from submitting their transactions, leading to the exclusion of valuable transactions from the blockchain and the allocation of blockspace to users with lower private values, ultimately reducing social welfare. Secondly, even if a frontrunnable user does submit their transaction, it can be front-run by attackers, resulting in a wealth transfer and taking up blockspace that could have been utilized by transactions with higher private values. The following example further illustrates these sources of inefficiencies.

Example. Suppose that the block capacity is $B = 3$, and there are four transactions with values $v_0 > v_1 > v_2 > v_3$. Let the first transaction be frontrunnable and the rest non-frontrunnable. The efficient allocation of blockspace, which maximizes aggregate welfare, is shown in Table 1a. However, this efficient allocation may not be achievable due to the two inefficiencies caused by frontrunning risk. Table 1b shows the inefficiency caused by transaction exclusion when frontrunning risk is severe ($p * c > v_0$). In this case, the frontrunnable transaction (v_0) will not submit, and blockspace will be allocated to transactions with lower values, v_1, v_2, v_3. Table 1c shows the inefficiency caused by blockspace waste where the frontrunning attack transaction takes up valuable blockspace. The attack transaction does not add value but merely transfers an amount c from the frontrunnable user, so the first two slots combined only contribute v_0 to the aggregate welfare. In this case, only two slots are allocated to economically meaningful transactions, and the aggregate welfare is only $v_0 + v_1$.

The following proposition formally characterizes the allocative inefficiency in the absence of a private pool:

Proposition 1 (Only Public Pool Benchmark)**.** *With only a public pool, there is a threshold $c_1 \geq 0$ such that:*

1. *If $c > c_1$, the frontrunnable user does not submit their transaction and the aggregate welfare in the equilibrium is $\sum_{i=1}^{B+1} v_i$.*

2. *If $c \leq c_1$, the frontrunnable user submits their transaction to the blockchain and it gets front-run by attackers, resulting in an aggregate welfare in the equilibrium of $\sum_{i=0}^{B-1} v_i$.*

(a) Efficient Allocation

Position	**Transaction**
1	v_0
2	v_1
3	v_2
Total	$v_0 + v_1 + v_2$

(b) Transacation Exclusion

Position	**Transaction**
1	v_1
2	v_2
3	v_3
Total	$v_1 + v_2 + v_3$

(c) Blockspace Waste

Position	**Transaction**
1	c
2	v_0 -c
3	v_1
Total	$v_0 + v_1$

Table 1: This table illustrates the two sources of blockspace allocation inefficiency. Table 1a shows the efficient allocation where the aggregate welfare is maximized. Table 1b illustrates the inefficiency due to transaction exclusion, which occurs when frontrunning risk is severe ($c > v_0$). In this scenario, the frontrunnable transaction is not submitted, and blockspace is allocated to transactions with lower values. Table 1c illustrates the inefficiency due to blockspace waste, where the first blockspace is taken by the attack transaction. In this case, only two slots are allocated to economically meaningful transactions.

This proposition shows that if the frontrunning problem is severe ($c > c_1$), it is not incentivized for the frontrunnable user to submit their transaction, as the cost of being front-run outweighs the benefit of executing the transaction. In this case, inefficiency arises ($\sum_{i=1}^{B+1} v_i < \sum_{i=0}^{B} v_i$) due to the exclusion of the transaction from the blockchain. On the other hand, if the frontrunning problem is not too severe ($c \leq c_1$), the frontrunnable user will still submit their transaction despite the frontrunning risk. In this case, inefficiency arises ($\sum_{i=0}^{B-1} v_i < \sum_{i=0}^{B} v_i$) because one blockspace is wasted on the frontrunning attack.

4.2 Pool choice of attackers

We analyze the equilibrium with the presence of the private pool by first considering the pool choice strategies of attackers. For any adoption rate α of the private pool, we only consider the case where the frontrunnable user chooses the public pool. This consideration suffices because if the frontrunnable user were to submit through the private pool, her transaction would be unobservable by attackers and they would not be able to front-run her. We denote $f^{(B)}$ as the B^{th} largest fee submitted by users in period 2.

In order to prioritize their execution, both attackers will use both the private and public pools. The private pool provides prioritized execution, as transactions submitted through it are placed at the top of the block by participating validators. However, relying solely on the private pool carries execution risk, as some validators may not observe transactions submitted through it. Thus, attackers also submit their transactions to the public pool to ensure execution. The following proposition characterizes the pool choice and fee-bidding strategies of attackers in equilibrium.

The following proposition characterizes the pool choice and fee bidding strategies of attackers in equilibrium:

Proposition 2 (Pool Choice and Fees Bidded by Attackers)**.** *In equilibrium, two attackers send transactions to both the public and private pools. Both attackers truthfully bid c in the private pool, as this secures their prioritized execution. However, in the public pool, their strategy depends on the value of f_0. If $f_0 < c - \epsilon$, one attacker will place an opening bid of $\max(f^{(B)}, f_0) + \epsilon$, and then increment their bid by the minimal amount ϵ in each subsequent bidding round. If $f_0 \geq c - \epsilon$, the attackers will not bid in the public pool.*

The different auction forms in the private pool and public pool also play a crucial role. In the private pool, the seal-bid, first-price auction form makes attackers bid their true value of the transaction, which is equal to their frontrunning profits. This results in high transaction fees being bid by attackers in the private pool.

In contrast, in the public pool, the pay-as-bid, first-price auction form leads to a different bidding strategy by attackers. They will always increase their bids slightly in each round of the auction, and will not bid their true value. The fees paid by attackers in the public pool will be lower than in the private pool, but still higher than the minimum required value to guarantee their transaction's execution by the validators.

As a result, the introduction of the private pool exacerbates the competition among attackers, leading to higher transaction fees being paid, which are then earned by the winning validator as part of their miner extractable value (MEV). This highlights the fact that validators benefit from the frontrunning, as they earn a portion of the MEV for each frontrunning opportunity.

4.3 Pool Choice of Users

The strategy of the frontrunnable user is analyzed based on the exogenously specified relay adoption rate α. The choice between the private pool and public pool presents a straightforward trade-off for the user: while the private pool eliminates the risk of frontrunning, it exposes the user to the risk of transaction execution. Unlike attackers, the user does not submit through the private pool to outbid competitors, but rather to avoid frontrunning. If the adoption rate of validators for the private pool is high enough, the execution risk is low, and the user will choose the private pool. The following proposition characterizes the user's pool choice strategy in equilibrium:

Proposition 3 (Pool Choice of Frontrunnable User)**.** *There is a critical threshold* $0 \leq \lambda \leq 1$ *such that the frontrunnable user will choose the private pool for her transaction if and only if* $\alpha > \lambda$.

4.4 Validators' Adoption and Equilibrium

We analyze the equilibrium adoption rate of the private pool by validators, $\alpha^* = \frac{M^*}{N}$, and characterize the SPE.

For any $\alpha = \frac{M}{N} > 0$, transactions submitted through the private pool can only be observed by validators who adopt it and may come with a high transaction fee. Thus, if the actions of users and attackers are fixed, each individual validator has an incentive to adopt the private pool:

$$r_{private}(\alpha) \geq r_{public}(\alpha). \tag{1}$$

However, not all validators may adopt the private pool in equilibrium. A high adoption rate of the private pool could reduce the expected payoff for validators as the frontrunnable user may route her transaction to the private pool if the execution risk is low, reducing frontrunning opportunities and MEV extracted by validators, and they may also receive fewer transaction fees from users due to less block space competition.

In equilibrium, validators in the private pool have no incentive to exit, that is,

$$r_{private}(\frac{M^*}{N}) \geq r_{public}(\frac{M^*-1}{N}) \text{ or } M^* = 0, \tag{2}$$

and validators in the public pool have no incentive to adopt the private pool, that is,

$$r_{public}(\frac{M^*}{N}) \geq r_{private}(\frac{M^*+1}{N}) \text{ or } M^* = N. \tag{3}$$

We characterize the SPE of our game with the following proposition. We refer to the equilibrium where the adoption rate of the private pool is $\alpha^* = 1$ as the *full adoption equilibrium*, the equilibrium where the adoption rate $\alpha^* \in (0,1)$ as the *partial adoption equilibrium*, and the equilibrium where the adoption rate $\alpha^* = 0$ as *no adoption equilibrium.*

Proposition 4 (Characterization of the Equilibrium)**.** *Let c_1 be the critical threshold identified in Proposition 1. The following statements hold for the SPE of the game:*

1. *There exists a full adoption equilibrium where $\alpha^* = 1$, and the frontrunnable user selects the private pool, while the attackers do not submit attack orders.*
2. *If $c \leq c_1$, there exists a partial adoption equilibrium where $0 < \alpha^* < 1$, the frontrunnable user submits her transaction through the public pool, and the attackers send their orders to both pools.*

The private pool will be, at least partially, adopted by validators, and the equilibrium outcome is contingent on the severity of the front-running problem.

Consider the case where the front-running problem is severe ($c > c_1$). According to proposition 1, without a private pool, the frontrunnable user is not incentivized to submit transactions to the blockchain due to the high front-running risk, leading to inefficiency. The introduction of a private pool provides the frontrunnable user with the option to submit her transaction privately, and in the unique SPE, all validators adopt the private pool in order to observe and earn the transaction fee from the frontrunnable user. This eliminates the inefficiency resulting from the exclusion of the frontrunnable user's transaction and results in an efficient block space allocation.

Now consider the scenario where the front-running problem is not too severe ($c \leq c_1$). As per Proposition 1, even without the private pool, the frontrunnable user would still submit her transaction to the blockchain despite the front-running risk, leading to inefficiency due to wasted block space. In this case, a partial adoption equilibrium exists, where only a fraction of validators adopt the private pool, resulting in high execution risk. To avoid the high execution risk, the frontrunnable user submits her transaction through the public pool, and front-running attacks persist. The inefficiency caused by wasted block space remains in the partial adoption equilibrium, resulting in an inefficient allocation even with the presence of the private pool.

Validators have an incentive to maintain the front-running attacks because it allows them to earn extra fees from the fee bidding war of attackers and from other users due to the increased demand for block space caused by the front-running attacks. The marginal validators who only monitor the public pool have no incentive to adopt the private pool and observe more transactions. This is because if an additional validator monitors the private pool, the execution risk at the private pool would become too low, leading users to submit transactions through the private pool instead of the public pool. This would effectively eliminate front-running and reduce the fees earned by all validators, lowering their revenue.

5 Welfare Implications

In this section, we examine the implications of the addition of a private pool to the public blockchain on overall welfare and investigate ways to enhance the current market design of the private pool to attain efficiency. Additionally, we assess the impact on the welfare of individual participants, blockchain congestion, and transaction fees.

5.1 Aggregate Welfare and Blockspace Allocation

Proposition 5 illustrates the impact of incorporating a private pool on aggregate welfare:

Proposition 5 (Aggregate Welfare)**.** *The following statements hold:*

1. *A private pool weakly increases aggregate welfare.*
2. *Aggregate welfare is maximized when all validators adopt the private pool.*
3. *If $c > c_1$, the unique full adoption equilibrium is socially efficient. If $c \leq c_1$, any partial adoption equilibrium is not socially efficient.*

This result can be understood as follows. By having both a private and a public pool, users have the option to send their transactions privately, thus eliminating the inefficiency caused by transaction exclusion and resulting in higher aggregate welfare. However, if only a fraction of validators adopt the private pool, inefficiency due to block space waste from front-running attacks remains in the partial adoption equilibrium.

The maximum aggregate welfare can only be achieved if front-running risk is fully mitigated. If all validators adopt the private pool, the frontrunnable user can submit her transaction privately without facing execution risk, thus eliminating front-running risk and blockspace waste, and achieving the socially optimal allocation of block space, where the block only includes the B users' transactions with the highest private values.

Inefficient partial adoption equilibrium occurs because the marginal validator does not want to forego the rent earned from MEV. Despite there being no investment required for validators to adopt the private pool, the cause of partial adoption equilibrium resembles the classic hold-up problem. Validators do not fully adopt the private pool (sellers under-invest) despite it being socially sub-optimal because the gains from adoption (investment) would be appropriated by the frontrunnable users (buyers), effectively reducing the validators' (selles') revenue.

This misalignment of incentives between validators and the front-runnable user can be resolved through a contract that requires users to pay an additional fee for each order executed through the private pool. This way, validators will be incentivized to adopt the private pool, leading to a full adoption equilibrium and maximizing aggregate welfare.

Proposition 6 (Attaining Full Adoption)**.** *There exists a value $\theta \geq 0$ such that if the frontrunnable user pre-commits to paying a fee θ to any validator who executes her transaction sent through the private pool, then:*

1. *A full adoption equilibrium is attained, and the aggregate welfare is maximized.*
2. *The expected payoff of all validators strictly increases.*
3. *The expected payoff of the frontrunnable user does not decrease.*

In the partial adoption equilibrium, the total loss suffered by the frontrunnable user due to front-running, c, is equal to the total MEV extracted by validators and attackers. The portion of the MEV earned by validators, m, is in the form of fees paid by attackers, and the rest, $c - m$, is captured by attackers. The frontrunnable user is willing to pay as much as c to eliminate front-running risk, and validators are willing to fully adopt the private pool if the benefit of doing so is greater than m.

By pre-committing to paying a fee $\theta > m$ to any validator who executes her trade sent through the private pool, the frontrunnable user incentivizes all validators to adopt the private pool, maximizing aggregate welfare. As long as $\theta < c$, the frontrunnable user's expected payoff increases.

This contract can be implemented through a reward pool, which is a smart contract that allows users to deposit tokens voluntarily. Validators who join the relay service and successfully mine a block including transactions sent through the relay can claim the tokens in the reward pool. Private pools can also charge users flat fees for each successfully executed transaction. In this way, frontrunnable users can pay for their pre-trade privacy and incentivize validators to adopt private pools.

5.2 Who Benefits from the Private Pool?

In this section, we analyze the impact of the private pool on the welfare of different parties involved in the blockchain, including users, validators and attackers. We define the welfare of validators as the total expected transaction fees paid by users and attackers, and the welfare of attackers as the sum of their expected payoffs.

To analyze the individual's welfare, we consider the equilibrium with the lowest validators' adoption rate of the private pool among all equilibria. This choice is based on the following reasoning: we start with the scenario where all validators are in the public pool. If some validators find it profitable to adopt the private pool, they will do so, and the migration of validators from the public to the private pool continues until a stable state is reached, where all validators in the public pool have no incentive to adopt the private pool, and all validators in the private pool are better off not leaving it.

Proposition 7 (Welfare of validators, user, and attackers)**.** *The introduction of a private pool increases the welfare of validators, reduces the welfare of attackers, increases the expected payoff of the frontrunnable user, and decreases the expected payoff of non-frontrunnable users $i, i = 1, ..., B-2$.*

The increase in welfare for validators can be attributed to two factors: an increase in MEV extracted, and an increase in transaction fees due to a higher demand for block space. The private pool exacerbates competition between attackers and increases MEV, as discussed in Proposition 2. This reduces the welfare of attackers as a higher portion of their profits is transferred to validators who adopt the private pool. Additionally, the private pool may incentivize the frontrunnable user to submit her transaction, thereby increasing the demand for block space and validators' revenue from fees.

The expected payoff of the frontrunnable user increases because she can now hide the content of her transaction. However, this increase is not strict at partial adoption equilibrium. On the other hand, the expected payoff of most non-frontrunnable users decreases as their transaction fees increase with the introduction of a private pool, as discussed in the following Proposition:

Proposition 8 (Transaction Fees)**.** *The option of using the private pool increases the minimum fee required to guarantee the execution of a transaction.*

The above proposition highlights that, despite increasing welfare, the introduction of a private pool may have the potential to increase blockchain congestion and transaction fees. One might expect transaction fees to decline as the private pool reduces the block space used by attackers, but our analysis shows otherwise. As indicated in part 1 of Proposition 4, the private pool may result in more frontrunnable transactions, driving up demand for block space and ultimately leading to higher transaction fees.

6 Empirical Analysis

In this section, we present empirical support for the predictions of our model. Section 6.1 outlines the key predictions of the model. In Section 6.2, we describe the data used in our analysis. Section 6.3 defines the key variables and presents stylized facts. Finally, in Section 6.4, we present our empirical results.

6.1 Testable Implications

Our model has made several predictions on the impact of private pools in public blockchains. These predictions include (1) the adoption of private pools by validators, attackers and users under certain conditions, (2) the changes in the welfare of market participants with the incorporation of private pools, and (3) the limitations of private pools in fully eliminating frontrunning and blockspace waste. The implementation of Flashbots private pools provides us with the opportunity to empirically verify the following predictions:

1. At least a fraction of validators will adopt the private pool (see Proposition 4), and validators utilizing the private pool will have higher expected payoffs than those remaining in the public

pool (See equation (1)).

2. Attackers adopt private pools, but their payoffs decrease due to intensified competition, as implied by Proposition 7.

3. Under high frontrunning risk, users are incentivized to submit transactions through the private pool (see Proposition 4).

4. The incorporation of a private pool into the public blockchain does not decrease inefficiency caused by blockspace waste (see Proposition 4).

6.2 Data

To support our empirical analysis, we utilize transaction-level data from Uniswap and Sushiswap to identify frontrunning attacks. We run our own Ethereum node and use a modified geth client to export all transaction receipts that triggered a *swap* event in the smart contract of either Uniswap or Sushiswap. Our dataset covers the period from block number 10000835 (created on May 4, 2020) to block number 12344944 (created on April 30, 2021) and includes transaction details such as slippage threshold, gas used, and gas price. We follow the method outlined in Wang et al. (2022) to identify frontrunning attacks and calculate their revenues.

In addition, we collect data on transactions submitted through the Flashbots private pool to validators via the API services provided by Flashbots. Our data collection period starts from February 11, 2021 (when the first Flashbots block was mined) and ends on July 31, 2021, to eliminate the influence of the new fee mechanism introduced by EIP 1559 after August 2021.

Finally, we source Ethereum block data from Blockchair (available at `https://gz.blockchair.com/ethereum/blocks/`) covering the period from May 1, 2020 to July 31, 2021, including the gas fee revenues earned by validators.

6.3 Definition and Stylized Facts of Key Variables

In this section, we outline the main variables used in our statistical analysis and present some empirical regularities observed in our dataset.

Adoption Rate of the Private Pool by Validators. We calculate the adoption rate of the

Flashbots private pool in day t as the ratio of the number of blocks mined in day t that contain Flashbots transactions and the total number of blocks mined in day t.

Validators' Revenue per Block. The revenue earned by validators per block is composed of the transaction fees paid by transactions submitted through both the private pool and the public mempool. Specifically, if a validator mines a block containing transactions submitted through the Flashbots private pool, the revenue includes the fees from these transactions as well as those from the public mempool. On the other hand, if a block only contains transactions submitted through the public mempool, then the revenue is solely derived from the transaction fees of these transactions. The fixed block reward is not included in our measure of validators' revenue.

Cost-to-Revenue Ratio for Frontrunning Attackers. In our analysis, we quantify the cost-to-revenue ratio for each frontrunning attack we identify by dividing the transaction fee paid by the attacker by the revenue generated from the frontrunning attack. The higher the cost-to revenue ratio, the more portion of frontrunning profits are appropriated by validators for each frontrunning attack. Both the transaction fee and the revenue are expressed in the unit of Ether.

Probability of being Frontrun for User Transactions. For each transaction submitted through the public mempool, we determine its frontrunnability and whether it has been frontrun using the methodology outlined in Appendix B.2. At a very high-level, we define a transaction as frontrunnable if it is profitable to frontrun and then backrun such transaction given the transaction fee paid this transaction, the trade size, and the slippage tolerance of this transaction. We identified a transaction as being frontrun if there exists an attacker who first frontrun this transaction and then backrun transaction and made a positive profit. The probability of being frontrun in the public pool in a given day t is calculated as the number of transaction that are actually frontrun by attackers in that day divided by the number of all frontrunnable transactions submitted to the public pool on that day.

Proportion of User Transactions Sent Through the Private Pool. For each transaction submitted through the Flashbots private pool, we examine whether it would be frontrunnable if submitted through the public mempool. The proportion of user transactions sent through the private pool in day t is calculated as the number of frontrunnable transactions submitted through Flashbots during day t divided by the number of all frontrunnable transactions submitted in both the private and public pool during that day.

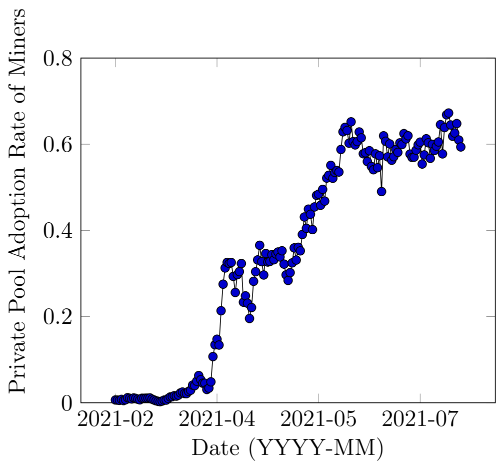


Figure 3: Adoption Rate of Flashbots Private Pool.

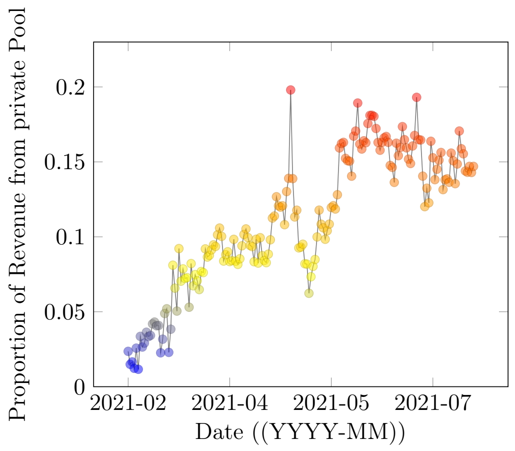


Figure 4: Proportion of Validators' Revenue Extracted from Flashbots Private Pool Transactions.

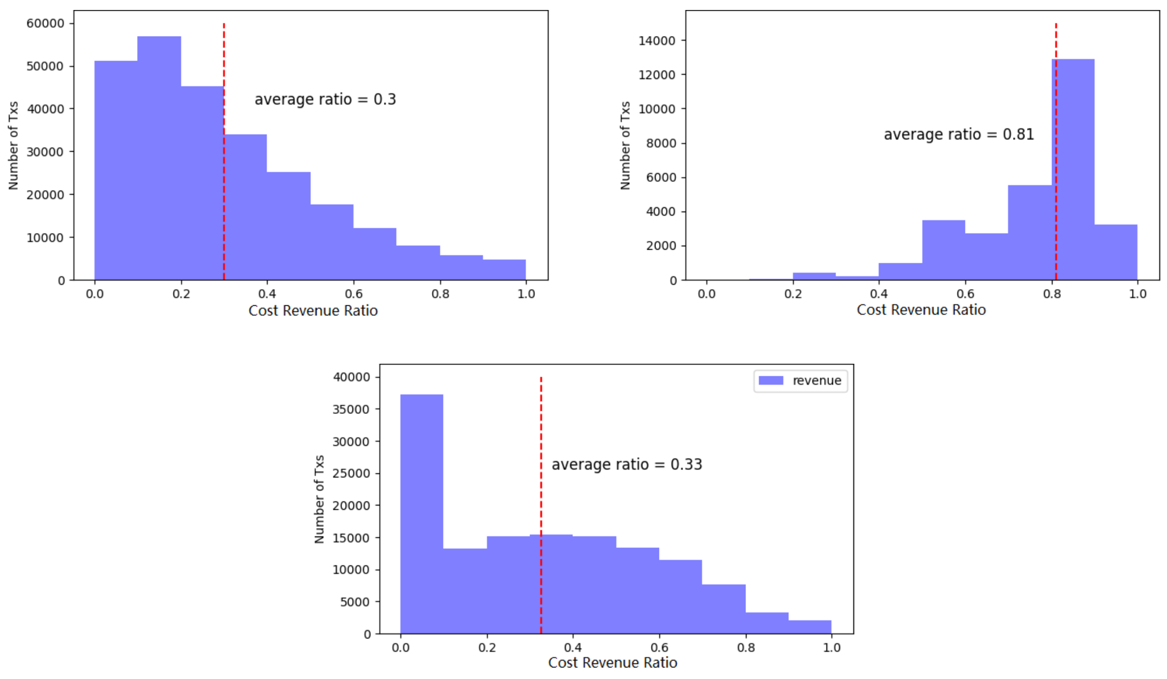


Figure 5: Panel (a) - top left: Distribution of the Cost-to-Revenue Ratio of Attackers in the Public Mempool Before the Introduction of Flashbots. Panel (b) - top right: Distribution of the Cost-to-Revenue Ratio of Attackers on Flashbots private pool. Panel (c) - bottom: Distribution of the Cost-to-Revenue Ratio of Attackers in the Public Mempool After the Introduction of Flashbots.

	N	Mean	SD	10th	50th	90th
Panel A: Validators Data						
Daily Private Pool Adoption Rate	171	0.343	0.239	0.01	0.346	0.613
Revenues of Validators Monitoring Private Pool (ETH)	377,366	0.972	17.82	0.235	0.606	2.2
Proportion of Revenue from Private Transactions (ETH)	377,366	0.139	0.148	0.024	0.086	0.326
Revenues of Validators Monitoring only Public Mempool (ETH)	2,582,015	1.161	9.585	0.231	0.832	2.36
Panel B: Attackers Data						
Frontunning Profit of Attackers Using Private Pool (ETH)	29,465	0.248	0.495	0.042	0.125	0.497
Transaction Cost of Attackers Using Private Pool (ETH)	29,465	0.182	0.363	0.032	0.092	0.371
Cost-to-revenue Ratio of Attackers Using Private Pool	29,465	0.755	0.151	0.51	0.801	0.901
Frontunning Profit of Attackers Using Public Pool (ETH)	394,239	0.204	0.571	0.033	0.091	0.408
Frontunning Profit of Attackers Using Public Pool (ETH)	394,239	0.04	0.093	0.004	0.023	0.069
Cost-to-revenue Ratio of Attackers Using Public Pool	394,239	0.309	0.239	0.021	0.261	0.662
Panel C: Users Data						
Daily Probability of Being Attacked for Users in Public Pool	80	0.165	0.034	0.120	0.165	0.209
Daily Proportion of Frontrunnable Transactions Sent to Private Pool	80	0.033	0.038	0	0.01	0.09

Table 2: Summary Statistics

Stylized Facts and Descriptive Statistics. Our data is summarized in Table 2, which provides a range of summary statistics. In Panel A, we examine the daily adoption rate of the private pool by validators and their resulting revenues. The estimated daily adoption rate of the Flashbots private pool is plotted in Figure 3. The average adoption rate by validators is approximately 35%, which supports our model's implication that the private pool is being partially adopted. For validators who participate in the Flashbots private pool, Figure 4 plots their proportion of revenue extracted from private pool transactions. It is evident that transaction fees from the private pool contribute a non-trivial portion (around 15%) of the revenues for validators who monitor it. As expected, the proportion of fee revenue from the private pool increases with its adoption rate, as low adoption by validators likely leads to low adoption by users. (See Proposition 3).

In Panel B of Table 2, we present the stylized facts regarding the profit and cost of attackers. The data reveals that roughly 30,000 frontrunning attacks were conducted through the Flashbots private pool, constituting approximately 20% of all frontrunning attacks (totaling around 170,000) during the analyzed period. It is noteworthy that, considering only 35% of blocks are appended by validators monitoring the private pool, even if attackers utilized both the private pool and public mempool equally for each transaction, only 35% of frontrunning attacks would have taken place through the private pool. This observation supports the conclusion that attackers indeed adopt the private pool.

To provide further insight, we plot the distribution of the cost-to-revenue ratios of attackers in Figure 5. A comparison of Panels (a) to (c) shows that the cost-to-revenue ratio for attackers executing their frontrunning activities through the Flashbots private pool is significantly higher compared to those using the public mempool. This is consistent with our prediction in 2 that attackers are willing to submit higher transaction fees in the private pool. Figure 6 displays the daily average cost-to-revenue ratio of attackers in both the public mempool and Flashbots private pool.

In Panel C of Table 2, we present data on users' pool choice and their estimated probability of being frontrun. Figure 7 plots users' probability of being frontrun in the public pool (red) and the proportion of their transactions submitted through Flashbots (black). The graph supports our model's prediction that higher frontrunning risks incentivize users to use the private pool.

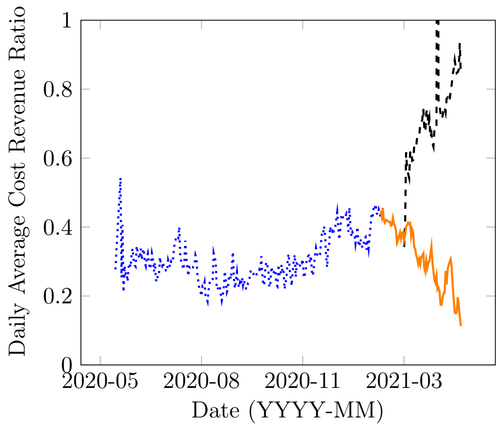


Figure 6: Daily Average Cost-to-Revenue Ratio of Attackers. The blue dotted line represents attackers in the public mempool before the introduction of Flashbots. The black dashed line represents attack transactions executed through the Flashbots private pool. The orange solid line represents attack transactions executed through the public mempool after the introduction of Flashbots.

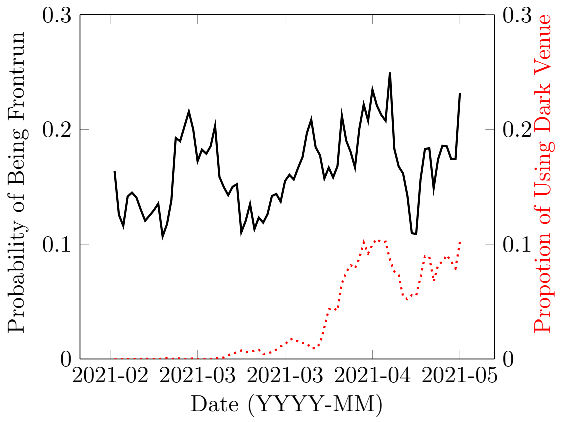


Figure 7: Black Solid Line: Daily Average Probability of a Frontrunnable Transaction Being Attacked in the Public Pool. Red Dotted Line: Daily Proportion of Frontrunnable Transactions Submitted Through Flashbots Private Pool.

6.4 Empirical Results

In this section, we present empirical evidence that supports the main testable implications of our model.

6.4.1 Validators' Revenue.

To assess the impact of participating in the Flashbots private pool on validators' revenue, we estimate the following linear model:

$$MinerRevenue_t = \gamma_t + \rho_1 \mathbb{1}_{Private} + \epsilon_t, \tag{4}$$

where t indexes the date, $MinerRevenue_t$ is the revenue of the miner per block, γ_t is the day fixed effects, $\mathbb{1}_{Private}$ is a dummy variable for blocks mined by validators participating in the Flashbots pool, and ϵ_t is an error term. Our standard errors are clustered at the day level. The coefficient ρ_1 measures the change in revenue per block after a miner joins Flashbots.

Table 3 shows the results of our regression analysis. The estimates indicate that, on average, validators who participate in the Flashbots private pool earn an additional 0.16 ETH per block compared to those who only monitor the public pool. This finding is in line with our model prediction that the expected payoff of validators who adopt the private pool is higher than that of validators who stay on the public pool. The coefficient estimates are statistically and economically significant.

Table 3: Results from regressing a binary variable indicating whether or not the miner of the block joins Flashbots on revenue from mining the block. The regression data covers the period from Nov 1, 2020 to July 31, 2021. Day fixed effects are included for all regressions. Standard errors are clustered at the day level. Asterisks denote significance levels (***=1%, **=5%, *=10%).

	Dependent variables: Validator's Revenue per Block
Intercept	1.21***
	(0.06)
Private	0.16***
	(0.032)
Day fixed effects?	yes
Observations	1,762,017
R^2	0.02

6.4.2 Intensified Fee Bidding War of Attackers

We estimate two linear models to compare the cost-to-revenue ratio of attackers before and after the introduction of the Flashbots private pool. The models are specified as follows:

Table 4: Results from regressing the cost-to-revenue ratio of attackers on a dummy variable equal to one after the introduction of Flashbots and zero otherwise, and another dummy variable equal to one if the attack order has been submitted through Flashbots and zero otherwise. The data covers the period from May 4, 2020 to Jul 31, 2021. Asterisks denote significance levels (***=1%, **=5%, *=10%).

	Dependent variables: Cost-to-Revenue Ratio	
	(a)	(b)
Intercept	0.300***	0.300***
	(0.001)	(0.001)
After	0.091***	0.013***
	(0.001)	(0.001)
Private		0.441***
		(0.002)
Observations	428,685	428,685
R^2	0.03	0.19

$$CostRevRatio = \rho_2 \mathbb{1}_{After} + \epsilon, \tag{5}$$

$$CostRevRatio = \rho_3 \mathbb{1}_{After} + \rho_4 \mathbb{1}_{Private} + \epsilon, \tag{6}$$

where $CostRevRatio$ is the cost-to-revenue ratio of attackers, $\mathbb{1}_{After}$ is a dummy variable equal to one after the introduction of Flashbots and zero otherwise, $\mathbb{1}_{Private}$ is a dummy variable equal to one for transactions submitted through Flashbots and zero for transactions sent through the public mempool, and ϵ is an error term. The coefficient ρ_2 quantifies the difference in the cost-to-revenue ratio of attackers before and after the introduction of Flashbots private pool, while ρ_4 quantifies the difference between the cost-to-revenue ratio of attackers who submit through the public mempool and attackers who send transactions through Flashbots private pool, after the introduction of Flashbots.

The results in Table 4 (a) demonstrate that the average cost-to-revenue ratio of attackers increased by approximately 0.09 after the introduction of Flashbots private pool, which represents a nearly a third increase compared to the average cost-to-revenue ratio prior to the introduction of Flashbots (around 0.3). This suggests that validators have appropriated a larger portion of the frontrunning profit, which is consistent with the predictions of our model.

The results in Table 4 (b) indicate that the average cost-to-revenue ratio for attackers who utilized the Flashbots private pool is 0.44 higher compared to those who only submitted their transactions through the public mempool. The cost-to-revenue ratio for attackers in the public pool, on the other hand, remains largely unchanged from before the introduction of the private pool. This implies that the increase in the cost-to-revenue ratio after the introduction of Flashbots is primarily due to the use of the private pool by attackers. The results also show that approximately 75% of the profits generated from these attacks are appropriated by validators, which is more than double the appropriation of attackers' profits in the public pool (around 30%). These findings are both statistically and economically significant.

Our regression results support the notion that the introduction of the private pool has intensified fee bidding wars between attackers and increased their costs. Attackers bid at higher levels in the private pool, which in turn reduces their welfare.

6.4.3 When will Users Choose Pivate Pool?

We estimate the following linear model to measure the relationship between users' probability of being frontrun and their choice of public versus private pools:

$$ProportionPrivate = \kappa FrontrunProb + \epsilon, \tag{7}$$

$ProportionPrivate$ is the proportion of frontrunnable transactions sent through Flashbots, $FrontrunProb$ is the probability of being frontrun for transactions sent through the public mempool venue, and ϵ is an error term. The coefficient κ quantifies the sensitivity of users' choice of pools (Flashbots versus public mempool) to the frontrunning risk faced by users.

Table 5 indicates that an increase in the probability of being frontrun in public pool is positively correlated (60% correlation) with a higher proportion of users' transactions sent through Flashbots. A 1% increase in the probability of being frontrun is associated with a 0.6% increase in the proportion of frontrunnable transactions submitted through Flashbots. The coefficient estimates indicate that these relationships are statistically and economically significant.

In summary, Table 5 supports our model implication that frontrunnable users migrate from the public to private pools when they face higher frontrun risk.

Table 5: Results from regressing the proportion of frontrunnable transactions sent through Flashbots on the probability of being frontrun. The data covers a sample period from Feb 11, 2020, to May 1, 2021. Asterisks denote significance levels (***=1%, **=5%, *=10%).

	Dependent variables: *Proportion of Transactions Through Flashbots*
Intercept	-0.066**
	(0.18)
Probability of Being Frontrun	0.605***
	(0.010)
Observations	80
R^2	0.3

7 Conclusion

The usage of private pools exposes users to execution risk, but also allow them to protect the content of their transactions which are sent privately to validators and not broadcast in the public mempool. We have shown that private pools are welfare enhancing because they increase the amount of blockspace assigned to value generating transactions, and thus result in a higher allocative efficiency. However, our analysis shows that private pools neither necessarily eliminate frontrunning risk nor reduce the transaction costs on blockchain. This is the case because rent-seeking validators are relucant to forgo their huge rents extracted from attackers. Existing empirical evidence is supportive of such prediction. In the period between May 2020 and April 2021, frontrunning attacks have generated MEV above 100 million USD, of which around 35% has been extracted by validators. This is also the reason why relay services are referred to as "Frontrunning as a service (FaaS)" instead of "Frontrunning prevention service". Essentially, a private pool provides an efficient, transparent platform for attackers to extract MEV, which exacerbates the arms race between attackers and redistributes higher MEV to validators. However, it has not achieved its intended purpose of eliminating or mitigating frontrunning.

We have presented a solution to achieve full adoption and eliminate frontrunning. Such a solution requires users to credibly commit to paying an additional fee to validators for executing their orders privately. Alternative solutions to the frontrunning problem would be to reshape the communication mechanism between nodes, such as encrypting the mempool, or change the transaction ordering mechanism, for example introducing fair sequencing of orders. Whether a contractual solution such as the one proposed in this paper would be more effective than a solution at the protocol level is left for future research.

References

Raphael Auer, Jon Frost, and Jose María Vidal Pastor. 2022. *Miners as intermediaries: extractable value and market manipulation in crypto and DeFi.* BIS Bulletins 58. Bank for International Settlements. https://ideas.repec.org/p/bis/bisblt/58.html

Yannis Bakos and Hanna Halaburda. 2021. *Tradeoffs in Permissioned vs Permissionless Blockchains: Trust and Performance.* Working Paper.

Bruno Biais, Christophe Bisière, Matthieu Bouvard, and Catherine Casamatta. 2019. The Blockchain Folk Theorem. *The Review of Financial Studies* 32, 5 (04 2019), 1662–1715.

Vitalik Buterin. 2015. A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM.

Andrea Canidio and Vincent Danos. 2023. *Commitment Against Front Running Attacks.* Working Paper.

Agostino Capponi, Sveinn Olafsson, and Humoud Alsabah. 2019. *Proof-of-Work Cryptocurrencies: Does Mining Technology Undermine Decentralization?* Working Paper.

Hao Chung and Elaine Shi. 2021. Foundations of Transaction Fee Mechanism Design. *arXiv preprint arXiv:2111.03151* (2021).

Lin William Cong, Zhiguo He, and Jiasun Li. 2020a. Decentralized Mining in Centralized Pools. *The Review of Financial Studies* 34, 3 (04 2020), 1191–1235.

Lin William Cong, Ye Li, and Neng Wang. 2020b. *Token-Based Platform Finance.* NBER Working Papers 27810. National Bureau of Economic Research, Inc.

Phil Daian. 2022. MEV for the Next Trillion, It's Time to Get Serious… | Flashbots. https://writings.flashbots.net/mev-for-the-next-trillion

P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. 2020. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy (SP)*. 910–927.

David Easley, Maureen O'Hara, and Soumya Basu. 2019. From mining to markets: The evolution of bitcoin transaction fees. *Journal of Financial Economics* 134, 1 (2019), 91–109.

Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. 2019. Sok: Transparent dishonesty: front-running attacks on blockchain. In *International Conference on Financial Cryptography and Data Security*. Springer, 170–189.

flashbots. 2021. Ethereum is a dark forest. `https://medium.com/@danrobinson/ethereum-is-a-dark-forest-ecc5f0505dfff` [Accessed May 25, 2021].

Jingxing (Rowena) Gan, Gerry Tsoukalas, and Serguei Netessine. 2021. Initial Coin Offerings, Speculation, and Asset Tokenization. *Management Science* 67, 2 (2021), 914–931.

Joshua Gans and Richard Holden. 2022. A Solomonic Solution to Ownership Disputes: An Application to Blockchain Front-Running. 4039499 (Feb 2022). `https://papers.ssrn.com/abstract=4039499`

Campbell R. Harvey, Ashwin Ramachandran, and Joey Santoro. 2021. *DeFi and the Future of Finance*. Working Paper.

Gur Huberman, Jacob D Leshno, and Ciamac Moallemi. 2021. Monopoly without a Monopolist: An Economic Analysis of the Bitcoin Payment System. *The Review of Economic Studies* 88, 6 (03 2021), 3011–3040.

Kose John, Thomas J Rivera, and Fahad Saleh. 2020. *Economic Implications of Scaling Blockchains: Why the Consensus Protocol Matters*. Working Paper.

Alfred Lehar and Christine A. Parlour. 2020. Miner Collusion and the BitCoin Protocol. 3559894 (Mar 2020). `https://doi.org/10.2139/ssrn.3559894`

Alfred Lehar and Christine A Parlour. 2023. *Battle of the Bots: Flash loans, Miner Extractable Value and Efficient Settlement*. Working Paper.

Satoshi Nakamoto. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Unpublished Manuscript.

Muriel Niederle, Deborah D. Proctor, and Alvin E. Roth. 2006. What Will Be Needed for the New Gastroenterology Fellowship Match to Succeed? *Gastroenterology* 130, 1 (Jan 2006), 218–224. `https://doi.org/10.1053/j.gastro.2005.10.058`

Andreas Park. 2021. *The Conceptual Flaws of Constant Product Automated Market Making*. Working Paper.

Chris Piatt, Jeffrey Quesnelle, and Caleb Sheridan. 2021. *Eden Network*. Unpublished Manuscript.

Julien Prat and Benjamin Walter. 2021. An Equilibrium Model of the Market for Bitcoin Mining. *Journal of Political Economy* 129, 8 (2021), 2415–2452.

Alvin E. Roth. 2008. What Have We Learned from Market Design? *The Economic Journal* 118, 527 (Mar 2008), 285–310. `https://doi.org/10.1111/j.1468-0297.2007.02121.x`

Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. 2004. Kidney Exchange*. *The Quarterly Journal of Economics* 119, 2 (May 2004), 457–488. `https://doi.org/10.1162/0033553041382157`

Alvin E. Roth and Xiaolin Xing. 1997. Turnaround Time and Bottlenecks in Market Clearing: Decentralized Matching in the Market for Clinical Psychologists. *Journal of Political Economy* 105, 2 (1997), 284–329. `https://doi.org/10.1086/262074`

Tim Roughgarden. 2021. Transaction fee mechanism design. *ACM SIGecom Exchanges* 19, 1 (2021), 52–55.

Ioanid Roşu and Fahad Saleh. 2021. Evolution of Shares in a Proof-of-Stake Cryptocurrency. *Management Science* 67, 2 (2021), 661–672.

Fahad Saleh. 2020. Blockchain without Waste: Proof-of-Stake. *The Review of Financial Studies* 34, 3 (07 2020), 1156–1190.

Christof Ferreira Torres, Ramiro Camino, et al. 2021. Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain. In *30th USENIX Security Symposium (USENIX Security 21)*. 1343–1359.

Ye Wang, Patrick Zuest, Yaxing Yao, Zhicong Lu, and Roger Wattenhofer. 2022. Impact and User Perception of Sandwich Attacks in the DeFi Ecosystem. In *ACM Conference on Human Factors in Computing Systems (CHI), New Orleans, LA, USA*.

Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. 2022. SoK: MEV Countermeasures: Theory and Practice. `https://doi.org/10.48550/ARXIV.2212.05111`

A Technical Results and Proofs

Proofs of Proposition 2. If the frontrunnable user does not submit to the blockchain or submit to the private pool, then the attackers will have no user to front-run, and thus they will not submit any transactions.

If the B^{th} largest bid submitted by users in period 2, $f^{(B)}$, is larger than or equal to $c - \epsilon$, then both attackers will not submit. This is because the cost of getting on the blockchain will be larger than the gain from executing a frontrunning attack.

We then consider the case where users submit to the blockchain through the public pool, and $f^{(B)} < c - \epsilon$. We also only consider the case where both attackers observe the frontrunnable user's transaction, otherwise, there will be no frontrunning in period 3. We first outline all six potential equilibrium outcomes for the pool selection of attackers. We then solve for the equilibrium transaction fee bidding strategies in all six cases. Finally, we solve for the equilibrium venue selection strategies of attackers.

There are six potential equilibrium outcomes for attackers' pool selection: (1) Both attackers choose the private pool; (2) One attacker chooses the private pool, and the other attacker chooses the public pool; (3) One attacker chooses the private pool, and the other attacker chooses both private and public pool; (4) One attacker chooses the public pool, and the other attacker chooses both public and private; (5) Both attackers choose the public pool; (6) Each attacker chooses both public and private pool.

Case 1: Both attackers choose the private pool. Since it is a seal-bid, first-price auction for a constant value c, both attackers bid c in equilibrium. Note that no matter how large the frontrunnable user bids in the public pool, transactions submitted through the private will always have priority execution when the block is appended by a validator monitoring the private pool. Consequently, the attackers can still successfully frontrun by bidding c even if the frontrunnable user bids $f_0 > c$.

Case 2: one attacker chooses the private pool, and the other attacker chooses the public pool. As there is no competition for execution in the private pool, the attacker will bid the lowest bid to get on chain $f^{(B)} + \epsilon$ in the private pool when he observes an attack opportunity.

In the public pool, the other attacker will bid $f_0 + \epsilon$ if $f^{(B)} \leq f_0 < c - \epsilon$, in order to outbid the frontrunnable user. However, if the transaction cost, $f_0 + \epsilon$, exceeds the gain from the attack, c, the attacker will not bid in the public pool. If $f^{(B)} > f_0$ the attacker will bid only $f^{(B)} + \epsilon$.

Case 3: One attacker chooses the private pool, and the other attacker chooses both the private and public pools. Both attackers bid c in the private pool. They bid truthfully in the private pool because this is a sealed-bid first-price auction, where both bidders have the same constant valuation of c. As for the attacker submitting to both pools, he will bid $f_0 + \epsilon$ in the public pool if $f^{(B)} < f_0 < c - \epsilon$, $f^{(B)} + \epsilon$ in the public pool if $f^{(B)} \geq f_0$, and not bid in the public pool if $f_0 > c - \epsilon$.

Case 4: one attacker chooses the public pool, and the other attacker chooses both private and public pool. We first consider the strategy of the attacker who submits transactions to both public and private pools. In the private pool, this attacker always bids $f^{(B)} + \epsilon$ due to the absence of competition.

In the public pool, the attackers' strategy is influenced by the value of f_0. One attacker will always submit an opening bid equal to $f_0 + \epsilon$ in the public pool if $v_{B-1} < f_0 < c - \epsilon$, and equal to $v_{B-1} + \epsilon$ in the public pool if $v_{B-1} > f_0$. No attacker will bid in the public pool if $f_0 \geq c - \epsilon$. If the auction lasts a single round, then the attacker who submits the opening bid wins the auction. If the auction lasts two rounds, then another attacker also joins in the auction. The later attacker will submit a bid ϵ higher than the previous bid.

Case 5: both attackers choose the public pool. If both attackers choose the public pool, their bidding strategy is the same as in Case 4.

Case 6: both attackers choose both public and private pools If both attackers choose both public and private pools, they all bid truthfully in the private pool. This is because the bidding mechanism is a sealed-bid, first-price auction where both attackers have the same valuation. In the public pool, they all use the same bidding strategy as in Case 4.

We then calculate the expected equilibrium payoff of each attacker for all six cases, and construct the following payoff matrices.

If $f^{(B)} \geq f_0$, the payoff matrices of both attackers are as follows:

A_1, A_2	Private	Public	All
Private	0, 0	$\alpha(c-(f^{(B)}+\epsilon))$, $(1-\alpha)(c-(f^{(B)}+\epsilon))$	0, $(1-\alpha)(c-(f^{(B)}+\epsilon))$
Public	$(1-\alpha)(c-(f^{(B)}+\epsilon))$, $\alpha(c-(f^{(B)}+\epsilon))$	$\frac{1}{2}(c-(f^{(B)}+2\epsilon))$, $\frac{1}{2}(c-(f^{(B)}+2\epsilon))$	$\frac{1}{2}(1-\alpha)(c-(f^{(B)}+2\epsilon))$, $\frac{1}{2}(c-(f^{(B)}+2\epsilon))(1-\alpha)+\alpha(c-(f^{(B)}+\epsilon))$
All	$(1-\alpha)(c-(f^{(B)}+\epsilon))$, 0	$\frac{1}{2}(c-(f^{(B)}+2\epsilon))(1-\alpha)+\alpha(c-(f^{(B)}+\epsilon))$, $\frac{1}{2}(1-\alpha)(c-(f^{(B)}+2\epsilon))$	$(\frac{1}{2}(c-(f^{(B)}+2\epsilon))(1-\alpha))$, $(\frac{1}{2}(c-(f^{(B)}+2\epsilon))(1-\alpha))$

In this case, it is easy to check that the unique symmetric nash equilibrium for this subgame is achieved when both attackers use both the private and the public pool. When both attackers use the private pool only, both of their payoffs are zero, and both attackers can deviate to either using the public pool only or using both pools and have a strictly better payoff. This is because $\alpha(c-(f^{(B)}+\epsilon)) > 0, \forall \alpha > 0$ and $(1-\alpha)(c-(f^{(B)}+\epsilon)) > 0, \forall \alpha < 1$. When both attackers use the public pool only, any attacker can profitably deviate to using both pools. This is because the payoff of using both pools is strictly higher:

$$\frac{1}{2}(c-(f^{(B)}+2\epsilon)) < \frac{1}{2}(c-(f^{(B)}+2\epsilon))(1-\alpha)+\alpha(c-(f^{(B)}+\epsilon))$$

When both attackers are using both pools, they do not have any profitable deviation.

If $f^{(B)} < f_0 < c-\epsilon$, the payoff matrices of both attackers are as follows:

A_1, A_2	Private	Public	All
Private	0, 0	$\alpha(c-(f^{(B)}+\epsilon))$, 0	0, 0
Public	0, $\alpha(c-(f^{(B)}+\epsilon))$	$\frac{1}{2}(c-(f_0+2\epsilon))$, $\frac{1}{2}(c-(f_0+2\epsilon))$	$\frac{1}{2}(1-\alpha)(c-(f_0+2\epsilon))$, $\frac{1}{2}(c-(f_0+2\epsilon))(1-\alpha)+\alpha(c-(f^{(B)}+\epsilon))$
All	$(1-\alpha)(c-(f_0+\epsilon))$, 0	$\frac{1}{2}(c-(f_0+2\epsilon))(1-\alpha)+\alpha(c-(f^{(B)}+\epsilon))$, $\frac{1}{2}(1-\alpha)(c-(f_0+2\epsilon))$	$(\frac{1}{2}(c-(f_0+2\epsilon))(1-\alpha))$, $(\frac{1}{2}(c-(f_0+2\epsilon))(1-\alpha))$

In this case, following the same procedure as the previous case, we can easily verify that the unique symmetric nash equilibrium for this subgame is achieved when both attackers use both the private and the public pool.

If $f_0 > c-\epsilon$, the payoff matrices of both attackers are as follows. It is easy to check that the symmetric Nash equilibria are achieved when both attackers use the private pool or when both attackers use both pools.

□

A_1, A_2	Private	Public	All
Private	0, 0	$\alpha(c-(f^{(B)}+\epsilon))$, 0	0, 0
Public	0, $\alpha(c-(f^{(B)}+\epsilon))$	0, 0	0, $\alpha(c-(f^{(B)}+\epsilon))$
All	0, 0	$\alpha(c-(f^{(B)}+\epsilon))$, 0	0, 0

Proof of Proposition **??**. If the frontrunnable user chooses the private pool, her optimal fee will be $v_B + \epsilon$, and her expected payoff is

$$\alpha(v_0 - (v_B + \epsilon)).$$

This is because she does not face frontrunning risk, so she only has to bid a sufficiently high fee to ensure that she will get on the chain. In this way, her optimal choice is to "barely" outbid the B^{th} user. Note that the optimal fee choice of user $i, i = 1, 2, ..., B-1$ will also be $v_B + \epsilon$.

We then show that if instead, the frontrunnable user chooses the public pool, her optimal fee will be $v_{B-1} + \epsilon$, and her expected payoff is

$$(v_0 - c * p - (v_{B-1} + \epsilon))).$$

If she chooses a fee below v_{B-1}, she would not get on the chain and have a lower payoff $-c * p$. If she chooses any fee f_0 in between $v_{B-1} + \epsilon$ and $c - \epsilon$, her payoff will be $(v_0 - c * p - f_0)) < (v_0 - c*p - (v_{B-1} + \epsilon)))$. If she chooses a fee above $c - \epsilon$, then her payoff will be $(v_0 - c*\alpha*p - f_0)) <= (v_0 - c*\alpha*p - c + \epsilon)) < v_0 - c*p - (v_{B-1} + \epsilon))$. This is because we assume $(1-p)*c > v_{B-1}$. The above argument shows that the optimal fee of the frontrunnable user in the public pool is $v_{B-1} + \epsilon$. Note that the optimal fee choice of user $i, i = 1, 2, ..., B-2$ will also be $v_{B-1} + \epsilon$.

Comparing the payoff in the two cases, that is, using public pool and using the private pool, we have that the frontrunnable user chooses the public pool if and only if $\alpha > \lambda = \frac{v_0 - c*p - (v_{B-1}+\epsilon))}{v_0 - (v_B + \epsilon)}$ □

Proof of Proposition 1. Suppose $\alpha = 0$. If the frontrunnable user submits through the public pool, then the payoff of the frontrunnable user is

$$v_0 - c * p - (v_{B-1} + \epsilon)).$$

The quantity above is positive if and only if $c < c_1 = \frac{1}{p}(v_0 - (v_{B-1} + \epsilon))$. If it is positive, then the frontrunnable user will submit her transaction. Otherwise, she will not submit to the blockchain.

□

Proof of Proposition 4. If $c > c_1$, the frontrunnable user will only use the private pool. This is because the payoff received from submitting through the public channel is $v_0 - c * p - (v_{B-1} + \epsilon)) < 0$, while the payoff from submitting through the private pool is $\alpha(v_0 - (v_{B-1} + \epsilon)) \geq 0$.

Validators who monitor the public pool only earn $r_{lit}(\alpha) = v_{B+1} + \epsilon + (B-1)(v_B + \epsilon)$ after validating a block. Consider a marginal validator with small mass $\frac{1}{N} > 0$ who migrates from the public to the private pool. The payoff of each such validator is $r_{private}(\alpha + \frac{1}{N}) = B(v_B + \epsilon) > v_{B+1} + \epsilon + (B-1)(v_B + \epsilon)$. In equilibrium, all validators monitor the private pool.

If $c \leq c_1$, we can show that $\lfloor N\lambda \rfloor$ is a equilibrium, and it is easy to verify that the other equilibrium is 1.

At $\lfloor N\lambda \rfloor$, consider a marginal validator with sufficiently small mass $\frac{1}{N} > 0$ monitoring the private pool. The payoff of each such validator is equal to $(B-1)(v_{B-1} + \epsilon) + c * p + (1-p)v_{B-1}$. If these validators migrate to only monitor the public pool, their payoff from the public pool is $(B-1)(v_{B-1} + \epsilon) + (v_{B-1} + 2\epsilon) < (B-1)(v_{B-1} + \epsilon) + c * p + (1-p)v_{B-1}$. Hence, there is no incentive for them to migrate.

Consider a marginal validator with sufficiently small mass $\frac{1}{N} > 0$ who only monitor transactions through the public pool. Their payoff is equal to $(B-1)(v_{B-1} + \epsilon) + (v_{B-1} + 2\epsilon)$. If they migrate to monitor transactions submitted through the private pool, their payoff is equal to $r_{private}(\frac{\lfloor N\lambda \rfloor}{N} + \frac{1}{N}) = B(v_B + \epsilon) < (B-1)(v_{B-1} + \epsilon) + (v_{B-1} + 2\epsilon)$. There is no incentive for them to migrate. This is because if $\alpha > \lambda$, the frontrunnable user migrates to the private pool, and there is no longer a frontrunning attack.

At $\lfloor N\lambda \rfloor$, the frontrunnable user still submits to the public pool as shown in Proposition **??**, and the attackers submit to both public and private pools as shown in Proposition 2. □

Proof of Proposition 8. If $c > c_1$, the frontrunnable trader does not submit transactions at all, thus the minimum fee that guarantees the execution of a transaction is v_B and the total fee of all transactions is $B \cdot (v_{B+1} + \epsilon)$. With a private pool, the execution fee increases to $(v_B + \epsilon)$, while the total fee increases to $B \cdot (v_B + \epsilon)$.

If $c \leq c_1$, the minimum fee that guarantees the execution of a transaction is always $(v_{B-1} + \epsilon)$.

□

Proof of Proposition 7. We compare the welfare of validators, frontrunnable users, and attackers separately, with and without a private pool. We first consider the case where $c < c_1$.

Without a private pool, the expected payoff of the frontrunnable user before the private pool is allowed is

$$v_0 - (v_{B-1} + \epsilon) - c$$

.

The expected payoff of the winning validator is

$$(v_{B-1} + \epsilon) * (B - 1) + (v_{B-1} + 2\epsilon)$$

.

The expected payoff of attackers is

$$\frac{1}{2}(c - (v_{B-1} + 2\epsilon))$$

.

The expected payoff of all non-frontrunnable users is

$$\sum_{i=1}^{B-2} v_i - (v_{B-1} + \epsilon) * (B - 2)$$

.

Then, we consider the welfare of different stakeholders in SPE.

When $\alpha^* = \lfloor \lambda N \rfloor$, the frontrunnable user selects the public pool and the attackers select both. The expected payoff of the frontrunnable user is $v_0 - (v_{B-1} + \epsilon) - c$. The payoff of the winning validator is $(v_{B-1} + \epsilon) * (B - 1) + c$ if she monitors the private pool. The expected payoff of the winning validator is $(v_{B-1} + \epsilon) * (B - 1) + (v_{B-1} + 2\epsilon)$ if she monitors the public pool only. The payoff of attackers is $(\frac{1}{2}(c - (v_{B-1} + 2\epsilon))(1 - \alpha))$. The expected payoff of all non-frontrunnable users is $\sum_{i=1}^{B-2} v_i - (v_{B-1} + \epsilon) * (B - 2)$.

We then consider the case where $c > c_1$. Without a private pool, the expected payoff of the frontrunnable user before the private pool is allowed is 0, as she will not submit her transaction to the blockchain. The expected payoff of the winning validator is

$$(v_{B+1} + \epsilon) * B$$

. The expected payoff of attackers is 0. The expected payoff of all non-frontrunnable users is

$$\sum_{i=1}^{B} v_i - (v_{B+1} + \epsilon) * B$$

.

Then, we consider the welfare of different stakeholders in SPE.

When $\alpha^* = 1$, the frontrunnable user selects the private pool. The expected payoff of the frontrunnable user is $v_0 - (v_B + \epsilon)$. The payoff of the winning validator is $(v_B + \epsilon) * B$. The payoff of attackers is 0. The expected payoff of all non-frontrunnable users is $\sum_{i=1}^{B-1} v_i - (v_B + \epsilon) * (B - 1)$. It is easy to verify that the payoff of validators and frontrunnable users indeed increases, while the welfare of other agents decreases.

$\square$

Proof of Proposition 5. The aggregate welfare of all stakeholders is the sum of the valuations of transactions included in the block.

If $c > c_1$, then the frontrunnable trader does not submit transactions without the option of a private pool. Therefore, the aggregate social welfare of stakeholders is $\sum_{i=1}^{B} v_i$. Because full adoption is the only equilibrium in this scenario, the aggregate social welfare will increase to $\sum_{i=0}^{B-1} v_i$ after allowing for a private pool.

If $c \leq c_1$, the expected aggregate social welfare of stakeholders without a private pool is $\sum_{i=0}^{B-2} v_i$. The expected aggregate social welfare of stakeholders if a private pool is allowed is still $\sum_{i=0}^{B-2} v_i$

Therefore, the possibility of a private pool weakly raises aggregate welfare in all equilibria.

If the private pool is fully adopted, the sum of valuations of transactions included in the block is $\sum_{i=0}^{B-1} v_i$, and the maximized aggregated welfare is achieved. If a private pool is only partially adopted, an attack transaction is included in the block, and the aggregate welfare is only $\sum_{i=0}^{B-2} v_i$.

□

Proof of Proposition 6. If $c > c_1$, there exists a unique full adoption equilibrium at which the aggregate welfare is maximized. The required payment is then zero.

If $c \leq c_1$, there exists a partial adoption equilibrium. At this equilibrium, the adoption rate of the private pool is $\alpha^* = \lambda$. At equilibrium, the expected attack loss of the frontrunnable user is c. c is also the sum of the expected attack revenues of the two attackers. The sum of expected transaction fees paid by two attackers is $(1 - \alpha^*)(v_{B-1} + 2\epsilon) + \alpha^* c$.

If the frontrunnable user commits to pay $\theta = c$ to the validator who executes her order in the private pool. When $(v_{B-1} + \epsilon) - (v_B + \epsilon)$ is sufficiently small, $r_{private}(\lambda + \frac{1}{N}) = B(v_B + \epsilon) + c > (B-1)(v_{B-1} + \epsilon) + (v_{B-1} + 2\epsilon) = r_{lit}(\lambda + \frac{1}{N})$. In this way, a marginal validator will move to monitor the private pool, and any partial adoption equilibrium does not exist. In addition, the frontrunnable user is not worse off after making the payment.

□

B Empirical Methodology

B.1 Frontrunning attacks

In this section, we explain the methodology used to identify frontrunning attacks. We identify a two-legged trade (T_{A1}, T_{A2}) as a frontrunning attack, and a transaction T_V as the corresponding victim transaction, if the following conditions are met:

1. T_{A1} and T_{A2} are included in the same block, and T_{A1} is executed before T_{A2}. T_{A1} and T_{A2} have different transaction hashes.

2. T_{A1} and T_{A2} swap assets in the same liquidity pool, but in opposite directions. The input amount for the swap in T_{A2} is equal to the output amount of the swap in T_{A1}. In this way, the transaction T_{A2} closes the position built up in the first leg T_{A1}.

3. T_V is executed between T_{A1} and T_{A2}. T_V swaps assets in the same liquidity pool as T_{A1} and T_{A2}. T_V swaps assets in the same direction as T_{A1}.

4. Every transaction T_{A2} is mapped to exactly one transaction T_{A1}.

There exists frontrunning attacks where T_{A1} and T_{A2} are placed in different blocks. However, attackers normally prefer to include T_{A1} and T_{A2} in one block to minimize inventory risk. Nonetheless, the above procedure allows us to find a lower bound for the number of frontrunning attacks. The revenue of a frontrunning attack is the difference between the output of T_{A2} and the input of T_{A1}, and the profit is the revenue minus the gas fee paid for these two transactions.

B.2 Frontrunnable Transactions

In this section, we provide that methodology to identify transactions vulnerable to frontrunning attacks. Observe that not all frontrunnable transactions are exploited by attackers.

There were $17,644,672$ transactions in the given time frame. The input token of $9,003,759$ of these transactions is ETH. We only focus on those transactions. This is because most attackers are bots, and only conduct attacks where ETH serves as input token. For each transaction, we calculate the optimal revenue that an attacker can attain by frontrunning this transaction. If the revenue is positive, then we identify this transaction as frontrunnable.

A swap transaction often has a slippage tolerance threshold m which specifies the minimum amount of output token to be received in the transaction. If the price impact of the frontrunning transaction T_{A1} is too large, the slippage tolerance threshold of the victim transaction T_V may be triggered and T_V will automatically fail. In this case, the attack will not be profitable. This is why we have to account for the slippage tolerance threshold for each swap transaction in our calculation. Formally, let v be the amount of input token specified in the victim transaction T_V, and m the minimum amount of output token to be received. Let x be the amount of input token swapped in the frontrunning transaction T_{A1}. Let r_1 and r_2 represent the liquidity reserves of input token and output token in the pool. The transaction fee in Uniswap and Sushiswap is 0.3%. The victim transaction will not fail if

$$\frac{v \cdot 0.997 \cdot (r_2 - \frac{x \cdot 0.997 \cdot r_2}{r_1 + 0.997 \cdot x})}{(r_1 + x) + 0.997 \cdot v} \geq m.$$

We solve the largest x that satisfies the above inequality. The result can be written as

$$maxInput_{A1}(r_1, r_2, v, m) = \frac{5.01505 \cdot 10^{-7} \cdot t}{\sqrt{m}} - 1.0015 r_1 - 0.4985 v,$$

where

$$t = \sqrt{9000000 r_1^2 m + 3976036000000 r_1 r_2 v - 5964054000 r_1 m v + 988053892081 m v^2}.$$

The $maxInput_{A_1}$ is the largest trade size of transaction T_{A1} such that T_V will not fail. We can then calculate the output amount in the second leg of the attack T_{A2} which closes the position built up in T_{A1}. T_V is frontrunnable if the constructed frontrunning attack yields a positive revenue. In total, we identify $3,612,343$ frontrunnable transactions with ETH as the input token.