Do Private Transaction Pools Mitigate Frontrunning Risk?

Agostino Capponi[1], Ruizhe Jia[1], and Ye Wang[2]

[1] Columbia University, 2960 Broadway, New York, USA
{ac3827,rj2536}@columbia.edu
[2] University of Macau, Avenida da Universidade, Macao, China
wangye@um.edu.mo

Abstract. Blockchain users who submit transactions through private pools are guaranteed pre-trade privacy but face execution risk. We argue that private pools serve the intended purpose of eliminating frontrunning risk, only if such risk is high. Otherwise, some validators may decide to avoid monitoring private pools to preserve rents extracted from frontrunning bots. Private pools intensify the execution arms race for bots, thus decreasing their payoffs and increasing validators' rents. The private pool option reduces blockspace allocative inefficiencies and raises aggregate welfare.

1 Introduction

Blockchain technology, initially introduced as the backbone for digital currencies and decentralized payment systems ([8]), has expanded to support a broad range of financial services through the development of smart contract technologies [2, 5, 6]. However, as blockchain systems evolve to support broader financial services, a key concern is the problem of frontrunning attacks and their negative impact on the efficiency of blockspace allocation.

In a typical public blockchain system, the transactions broadcast through the peer-to-peer network are visible to any node on the network. By observing these pending transactions, malicious attackers can execute frontrunning attacks [9, 3], which have already generated a wealth transfer from victim users over 200 million USD [10]. These value transfers are also referred to as maximal extractable value (MEV) [4]. As frontrunning transactions do not generate any value, they only result in a waste of blockspace and thus reduce allocative efficiency.

Most recently, private transaction submission pools have been proposed as a solution to mitigate frontrunning [1]. Private pools are implemented with relay services such as Flashbots and Eden Network, which create a private submission channel where pending transactions are not publicly observable [13]. As a result, attackers cannot frontrun them.

We investigate the welfare impact of private submission pools on a public blockchain. We first propose a game-theoretic model to analyze whether a private pool would be adopted by participants. Our model features three types of agents,

namely validators, users, and attackers; and two pool types, namely a private and a public mempool. Validators decide whether or not to monitor the private pool in addition to the public pool. Heterogeneous users submit transactions to the blockchain either through the public or private pool. Users earn private value from executing their transactions on the blockchain. Attackers compete to frontrun transactions sent through the public pool by users.

We then examine whether it would achieve its intended purpose of mitigating front-running attacks and improving welfare. We show that, in equilibrium, private pools are adopted at least in part by validators, and in full by attackers. Even though private pools neither eliminate frontrunning attacks nor reduce transaction fees, we demonstrate that they reduce allocative inefficiencies and thus raise aggregate welfare. The welfare-maximizing outcome is achieved if all validators monitor the private pool, and thus no block space is allocated to frontrunning attackers. However, this outcome may not be attainable in equilibrium because validators have a strong incentive to preserve rents extracted from attackers and frontrunnable users.

2 Institutional Details of Relay Services

In this section, we discuss the "built-in" information leakage problem of blockchain, and the principles of relay services.

2.1 Blockchain and Information Leakage

A blockchain is a decentralized ledger maintained by nodes distributed over a P2P network. Every node can issue transactions and broadcast them to every node in the P2P network. Validators collect transactions into blocks, and append blocks to the existing chain. Users attach an upfront fee to their submitted transactions. For many public blockchains, including Ethereum, fees allow users to gain execution priority, in the sense that validators execute transactions in decreasing order of fees.

Every node in the blockchain network may observe pending transactions. This transparency is not of concern for payment transactions, because there is no gain to be made from frontrunning a payment transaction. However, information leakage becomes worrisome for DeFi transactions executed through smart contracts. Frontrunning attacks can then be very costly for users [12, 10]. Frontrunning includes displacement, insertion, and suppression attacks [11]. In a displacement attack, an attacker observes a profitable transaction from a victim user. She then broadcasts an identical transaction, but with a higher transaction fee. This frontrunning transaction will then be executed before the victim transaction. The attacker will earn the profit, while the victim transaction would fail. In an insertion attack, an attacker observes a transaction, say the buy order of a token in a decentralized exchange (DEX) from a victim user. She then broadcasts two transactions: a frontrunning transaction with a higher fee than the victim transaction and a backrunning transaction with a lower fee.

The frontrunning transaction buys the same token as the victim transaction, which results 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 again goes up due to price impact. The backrunning transaction then closes the position by selling this token. The selling price is higher than the purchase price in the frontrunning transaction, which results in a wealth transfer from the victim user to the attacker.In a suppression attack, an attacker observes a transaction from a victim user. She then broadcasts transactions with a higher fee in order to prevent the victim transaction from being included in the block. Insertion frontrunning attacks are currently the most common in DeFi [11].

2.2 Relay Services

Relay services are an implementation of private pools. A centralized relay service receives transactions from users and forwards them to validators, without broadcasting on the P2P network. Therefore, users' transactions cannot be observed by malicious attackers. The relay platform screens validators before they join and monitor their activities to ensure that they do not exploit observed information. [3] However, this still requires users to trust the centralized relay which monitors all the transactions sent through it. Only 8 out of more than two million transactions submitted through private pools were frontrun by attackers [14]. This validates the frontrunning protection of private pools.

The first relay service, Flashbots, was launched in January 2021. Validators who join Flashbots prioritize the highest bidding transactions submitted through the Flashbots relay by including them at the top of a block. The execution order of transactions is typically determined by a one-round, seal-bid, first-price auction. Hence, the submitter neither knows the transactions submitted by other users nor the attached fees. By contrast, transaction fee bidding in the public pool takes the form of an ascending price auction and may consist of multiple rounds of bid submissions. Moreover, pending transactions and corresponding fees are publicly observable.

After the merge, i.e., the transition of the Ethereum blockchain from PoW to PoS, the implementation of relay services and private pools changed slightly. The new implementation is called proposer-builder separation (PBS) [7]. First, users send their transactions privately to block builders. Builders pack transactions they collect from the private channel as well as the public mempool into a block, and forward it to the validators . Again, only validators who adopted the relay service can receive this block. Second, users no longer have to trust validators for not frontrunning them. This is because validators have to sign a blinded block (which only includes block headers) and passed it to the builder to fill in the block content. This means that validators commit to propose the block constructed by the builders of relay services. In this way, the role of building a block is delegated from the validators to the builders. Figure 1 depicts the different implementations of private pools before and after the merge.

[3]The Flashbots Fair Market Principles (FFMP) can be seen from https://hackmd.io/@Flashbots/fair-market-principles.

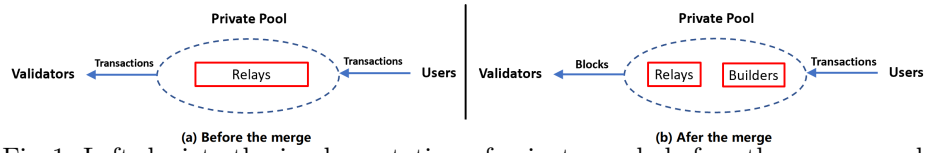

Fig. 1: Left depicts the implementation of private pools before the merge, and right illustrates the implementation of private pools after the merge.

3 Model Setup

The timeline consists of three periods indexed by t, $t = 1, 2, 3$. There are three types of agents: blockchain users, attackers, and validators. All agents are risk-neutral.

Validators. There are N homogeneous, rational validators. They all have the same probability, $\frac{1}{N}$, of earning the right to append a new block to the chain. At the end of period 3, the validator who appends the next block is drawn randomly from a uniform distribution. We assume that N is sufficiently large. This winning validator earns the fees attached to the transactions included in the block. Due to limited capacity, a validator can include at most B transactions in a block.

There exist two transaction submission channels: public and private pool. In period 1, validators can choose to adopt the private pool at no cost. We denote by M, and $\alpha = \frac{M}{N}$ the number and the portion of validators who adopt the private pool in period 1. All validators can view transactions submitted through the public pool, but only validators within the private pool can view transactions submitted through the private pool. Therefore, validators who join the private pool are able to monitor transactions from both the public and private pools. The transactions that are submitted to the private pool are not revealed to anyone other than the validators who adopted the private pool. We abstract away from the differences in implementation between different relays and between pre- and post-merge.

At the end of period 3, the winning validator selects the B transactions with the highest attached fees among those he observes. We assume that any tie will be broken uniformly at random. The validator decides the execution order as follows. If he has joined the private pool, then he prioritizes the transactions submitted through the private pool and executes them first. Those transactions will be executed in decreasing order of bid fees. Subsequently, the winning validator will include the transactions submitted through the public pool, again in decreasing order of fees. A validator who has not joined the private pool would add to his mined block the transactions submitted through the public pool only in decreasing order of fees.

Since a validator's adoption decision does not affect his probability of mining the next block, a validator decides whether to adopt a private pool to maximize the expected transaction fees conditional on him successfully mining 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 choice

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 the transactions.

The first type of user is one whose pending transaction can be frontrun, if submitted through the public pool and identified by attackers. We refer to this user as frontrunnable, and to her transaction as a frontrunnable transaction. The frontrunnable user gets a private value v_0 from executing her transaction. However, with probability p, attackers recognize that the pending transaction is frontrunnable, then they can front-run it and earn a profit $c \geq 0$. This, in turn, yields a loss of c to the front run user.

The second type of users are those whose transactions are not frontrunnable, even if they are broadcast through the public pool. We refer to this type of users as the non-frontrunnable users, and to their transactions as non-frontrunnable transactions. There exist $B+1$ non-frontrunnable users, indexed by $i \in \{1, 2, ..., B+1\}$, who extract private values $v_i, i \in \{1, 2, ..., B+1\}$ from their transactions. Without loss of generality, we let $v_1 > v_2 > ... > v_{B+1}$, and $v_0 > v_{B-1}, (1-p)c > v_{B-1}$. We also impose the following technical assumption to rule out corner cases:

Assumption 1 *The difference $v_{B-1} - v_B$ is sufficiently small.*[4]

In period 2, users decide which pools to send their transactions to. A user can broadcast her transaction through the public pool, through the private pool, or choose to not submit her transaction. If a frontrunnable transaction is broadcast through the public pool, it will face the risk of being identified and frontrun by attackers. If instead a transaction is only broadcast through the private pool, then it will not be observed by validators who do not monitor the private pool. Its probability of being included in the next block is at most α. Hence, the execution risk is determined by the validators' adoption rate of the private pool. We index the frontrunnable user as user 0. We denote the pool chosen by user $i, i \in \mathcal{I} = \{0, 1, 2, ..., B+1\}$, by $C_i \in \{\text{Private}, \text{Public}, \text{None}\}$. User i also attaches a transaction fee f_i to her transaction.

User i chooses her 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 the indicator function for the event "transaction by user i is included in the block by validator", $\mathbb{1}_{\text{frontrun,i}}$ is the indicator function for the event "transaction by user i is frontrun by attackers". We assume that users break

[4] If there are sufficiently many transactions, i.e., B is sufficiently large, and the private values that users earn 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.

any tie in favor of the public pool. Our assumption is justified by the fact that it usually requires more sophistication to submit transactions through private pools such as Flashbots Protect, and the interface for the public mempool is generally easier to use.

Attackers. There are two competing attackers, indexed by $j \in \mathcal{J} = \{1, 2\}$. The attackers first screen for the frontrunnable transaction in the public pool and then exploit it. The attack yields a profit $c \geq 0$. Any frontrunnable transaction pending in the public pool will be identified by both attackers with probability p.

In period 3, each attacker decides which pools to submit his order if he has recognized a frontrunnable transaction: public, private, or both. We denote the pool chosen by attacker j by $V_j \in \{\text{Public}, \text{Private}, \text{Both}\}$. We denote the transaction fee bid by attacker j in the private pool by f_{D_j}, and in the public pool by f_{L_j}. attacker j chooses the fee and pool to maximize his 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 "the order by attacker j is executed before the order by the other attacker", and $f_{executed,j}$ is the transaction fee paid by attacker j. The tie-break rule for attackers is that "both pools" is their preferred choice, the "public pool" is their second preferred choice, and the "private pool" is their last choice.

Transaction Fee Bidding. The attacker who bids the highest fee executes the order. The transaction fee bidding mechanisms in the public and private pools are different. Transaction fee bidding in the public pool, commonly referred to as Priority Gas Auctions (PGA), is a variant of an English Auction, i.e., an open-outcry ascending-price auction. Their main difference is that the transaction fee bidding in the public pool will end randomly when the validator decides the new block and stops receiving new transactions.

In period 3, if both attackers submit their transactions to the public pool, then there are either one round or two rounds of bidding, with the same probability. The first mover will be either of them with the same probability. In each round, only one attacker moves, and the bid increment has to be larger than ϵ. All bids in the public pool are visible to both attackers. To minimize downside risk, attackers deploy a smart contract. The smart contract would terminate the transaction if the attack no longer exists. In this case, the transaction would be 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, seal-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, then only 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}. We impose a symmetry restriction that the strategy used by both attackers are identical.

4 Model Analysis

In this section, we solve for the SPE of the game. We begin by analyzing the choice of the pool for attackers and users. Subsequently, We study the equilibrium adoption rate of the private pool, and the corresponding welfare implications.

4.1 Pool Choice of attackers

We analyze attackers' pool selection strategies, for any adoption rate α of the private pool, and assuming that the frontrunnable user chooses the public pool. Note that it suffices to consider this pool choice of the frontrunnable user only, because if she were to submit through the private pool, her transaction would not be observable by attackers. Hence, attackers would not be able to submit any attack order at $t = 3$. We denote $f^{(B)}$ as the B^{th} largest fees submitted by users in period 2.

In order to execute their orders before their competitors, both attackers will adopt private pools in addition to the public pool. This is because if an attacker chooses the private pool, then this attacker gains prioritized execution, since transactions submitted through the private pool are placed at the top of the block by validators who join this pool. However, using the private pool alone is not enough. Using the private pool only presents execution risk because a fraction of the validators may never observe transactions submitted through the private pool. In this way, attackers also submit their transactions to the public pool to guarantee execution. The following proposition characterizes the pool choice and fee-bidding strategies of attackers in equilibrium.

Proposition 1 (Pool Choice and Fees Bid by Attackers) *Attackers send transactions to both the public and private pools in equilibrium. Both attackers bid c in the private pool. In the public pool, if $f_0 < c - \epsilon$, one of the attackers places an opening bid $\max(f^{(B)}, f_0) + \epsilon$. Afterward, in each bidding round the attacker increases by the minimal increment ϵ from the previous highest bid. If $f_0 \geq c - \epsilon$, then the attackers will not bid in the public pool.*

Recall that transactions sent through the private pool will be guaranteed to be prioritized by validators who adopt this pool. To gain this benefit in the "arms

race" for priority execution, both attackers participate in the seal-bid, first-price auction in the private pool and bid truthfully, that is, bid transaction fees equal to their profits. In contrast, during each round of the PGA in the public pool, an attacker will only bid slightly higher than the previous highest bids. In this way, the private pool exacerbates the "arms race" among attackers and increases their fees paid.

Observe that the fee paid by attackers is pocketed by the winning validator. Because of competition, the fees bid by attackers is always higher than v_{B-1}, that is, the minimum fee which guarantees a transaction to be executed by validators. This suggests that validators extract a portion of MEV.

4.2 Pool Choice of Users

We analyze the selection strategy of the frontrunnable user, for an exogenously specified relay adoption rate α.

The main trade-off faced by the frontrunnable user is straightforward. Using the private pool exposes her to execution risk but eliminates the risk of being frontrun. Unlike attackers, the frontrunnable user does not submit through the private pool to outbid competitors but merely to avoid frontrunning. If the private pool's adoption rate of validators is sufficiently large, the execution risk is small, and the user will adopt the private pool. The following proposition characterizes her strategy in equilibrium:

Proposition 2 (Pool Choice of Users) *There exists a critical threshold* $0 \leq \lambda \leq 1$ *such that the frontrunnable user sends her transaction through the private pool if and only if* $\alpha > \lambda$.

4.3 Validators' Adoption and Equilibrium

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

For any $\alpha = \frac{M}{N} > 0$, the validators who adopt private pool receive a higher payoff than those who only stay on the public pool:

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

This is because transactions submitted through the private pool can only be observed by validators who adopt it. As a result, if the actions of users and attackers are fixed, each individual validator has an incentive to adopt the private pool.

However, it does not necessarily mean that all validators will adopt the private pool. The situation changes once we account for the strategic responses of users and attackers. If sufficiently many validators join the private pool, that is, if α is large enough, then the payoff of each validator may be lower than their payoff when $M = 0, \alpha = \frac{M}{N} = 0$. This is because the frontrunnable user may then route her transaction from the public to the private pool if the execution

risk in the private pool is small enough. The migration of this transaction would eliminate frontrunning opportunities and thus reduce MEV.

In equilibrium, the validators already in the private pool have no incentive to exit the private pool, 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 first characterize the equilibrium strategy of the frontrunnable user in the benchmark case where there is no private pool. This is obtained from our game theoretical framework by setting $\alpha = 0$, and considering the subgame at periods $t = 2, 3$.

Proposition 3 (Only Public Pool Benchmark) *If $\alpha = 0$, there exists a threshold $c_1 \geq 0$ such that the frontrunnable user submits the transaction to the blockchain if and only if $c \leq c_1$.*

If the frontrunning problem is severe, i.e., $c > c_1$, then it is not incentive compatible for the frontrunnable user to submit her transaction to the blockchain, because the cost of being frontrun exceeds the benefit of executing her transaction. Conversely, if the frontrunning problem is not too severe, i.e., $c \leq c_1$, then the frontrunnable user submits to the blockchain even if she faces the risk of being frontrun.

We next characterize the SPE of our game. We refer to the equilibrium where the adoption rate of 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 3. The following statements hold for the SPE of the game:*

1. *there exists a full adoption equilibrium where the adoption rate $\alpha^* = 1$, the frontrunnable user selects the private pool, and the attackers do not submit attack orders.*
2. *If $c \leq c_1$, there also exists a partial adoption equilibrium where the adoption rate of the private pool is $\alpha^* < 1$, 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. Suppose the frontrunning problem is severe ($c > c_1$). Recall from proposition 3 that

without a private pool, it is too costly for the frontrunnable user to submit transactions to the blockchain. Thus, she will only submit to the private pool. In equilibrium, all validators decide to adopt a private pool so that they are able to observe the transaction submitted by the frontrunnable user and earn her transaction fee.

Suppose the frontrunning problem is not too severe ($c \leq c_1$). Recall from proposition 3 that even without the private pool, the frontrunnable user would still submit her transaction to the blockchain even if she bears the risk of being frontrun. In such a case, partial adoption equilibrium will exist. Note that validators always extract a portion of MEV through the fees paid by attackers. To maintain MEV and keep the frontrunnable user in the public pool, only a fraction of validators choose to adopt the private pool in the partial adoption equilibrium, which creates high execution risk. As a result, the frontrunnable user prefers to submit through the public pool and face frontrunning risk. In such a case, the private pool does not prevent frontrunning attack. It may seem surprising why in the partial adoption equilibrium, validators who only monitor the public pool have no incentive also to monitor the private pool and observe more transactions. The reason is that if an additional validator monitors the private pool, then the execution risk at the private pool would be too low, users would submit transactions through the private pool instead of the public pool, and frontrunning risk would be eliminated. This would get rid of MEV and lower the revenue of all validators, including the marginal validator monitoring the public pool.

5 Welfare Implications

In this section, we analyze how the private pool impacts transaction fees on blockchain and agents' welfare. For our analysis in section 5.1 and 5.2, we select the equilibrium corresponding to the lowest validators' adoption rate of the private pool among all equilibria. [5] In section 5.3, we compare the blockspace allocation efficiency and welfare implications of different equilibria, and we propose a solution for always achieving the welfare-maximizing equilibrium.

5.1 Transaction Fees on Blockchain

We begin by showing that a private pool does not serve its intended purpose of reducing blockchain congestion and transaction fees.

Proposition 5 (Transaction Fees) *The option of using the private pool increases the minimum fee that guarantees the execution of a transaction.*

[5] We select according to the following rationale: consider the situation where all validators are in the public pool. Then some validators may find it profitable to adopt the private pool. Migration of validators from the public to the private pool continues until a stable state is reached, where all validators on the public pool have no incentive to adopt the private pool, and all validators on the private pool are better off not leaving it.

Because a private pool weakly reduces the block space used by attackers, one would expect a decline in transaction fees. Our analysis shows that this is not the case. As shown in part 1 of Proposition 4, a private pool may generate more frontrunnable transactions. This drives up the demand for block space and consequently results in higher transaction fees. It is crucial to recognize that higher transaction fees and increased demand for block space are not inherently detrimental to aggregate welfare. The existence of a private pool alleviates the friction of frontrunning, which can enhance the demand from users who place a higher value on their transactions. This improvement in allocative efficiency contributes positively to the overall welfare of the system.

5.2 Who Benefits from the Private Pool?

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.

Proposition 6 (Welfare of validators, user, and attackers) *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 decomposed into two parts: an increase in MEV extracted, and an increase in transaction fees due to a higher demand for block space. First, recall from Proposition 1 that a private pool exacerbates competition between attackers and increases MEV. This, in turn, reduces welfare for attackers, because a higher portion of their profits is transferred to validators who adopt the private pool. Second, recall that a private pool may incentivize the frontrunnable user to submit her transaction and thus increase the demand for block space. This, in turn, increases validators' revenue from fees.

The payoff of the frontrunnable user increases because she has can now hide the content of her transaction. It is worth observing that her payoff does not necessarily increase strictly. Unless the frontrunning problem is very severe, validators do not all adopt the private pool and this creates execution risk. As a result, the frontrunnable user may still find it preferable to submit through the public pool and bear frontrunning risk. In such case, her payoff stays unchanged and payoffs of most non-frontrunnable users decrease. Recall from Proposition 5 that their transaction fees increase with a private pool.

5.3 Aggregate Welfare and Blockspace Allocation

We analyze aggregate welfare, defined as the sum of expected payoffs of validators, users, and attackers. Note that the profit of attackers and fee revenue of validators are merely transfers of wealth from users, and transaction fees paid by attackers to validators are only part of the profits that attackers extract from

users. As a result, aggregate welfare is the sum of the private values of users' transactions added to the block.

With a public pool only, blockspace allocation is inefficient for two reasons. First, the frontrunnable user may not have enough incentive to submit her transaction because of high frontrunning risk. Blockspace is then allocated to users who earn less private value, resulting a social waste. Second, even if the frontrunnable user does submit her transaction, attackers will front-run her. The attack order is a wealth transfer but takes up blockspace that might have been used by transactions yielding private values to users. This is, again, socially inefficient. We provide an example of the blockspace allocation inefficiency in Appendix A.

The following proposition illustrates how introducing private pool reduces both types of inefficiencies and improve welfare:

Proposition 7 (Aggregate Welfare) *The followings statements hold:*

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

The above result can be intuitively understood as follows. With both a private and a public pool, users can send their transactions privately, and blockspace usage by attackers weakly decreases. Thus, both types of inefficiencies are reduced, and higher aggregate welfare is attained. The maximum aggregate welfare can only be achieved if frontrunning risk is mitigated. If all validators adopt the private pool, the frontrunnable user can submit her transaction privately without facing execution risk. As a result, frontrunning risk is eliminated, no attacker demands block space, and the block only includes the B users' transactions with the highest private values. Hence, aggregate welfare is maximized.

With the private pool, the social optimum is attained in equilibrium if the frontrunning problem is severe. Otherwise, the ecosystem may reach a partial adoption equilibrium where frontrunning attack still occurs, and social waste due to blockspace misallocation is not mitigated. This inefficient blockspace allocation occurs because the marginal validator does not want to forgo the rent from MEV. Although there is no investment required for validators to adopt the private pool, the cause of partial adoption equilibrium is still reminiscent of the classical hold-up problem: sellers under-invest (validators do not fully adopt) in the first stage, despite it being socially optimal, because the gains from the investment are appropriated by the buyers (front-runnable users).

This misalignment of incentives between validators and the front-runnable user can be resolved if both parties can sign a contract requiring users to pay an additional fee for each order executed through the private pool. In this way, validators are willing to adopt the private pool, and the resulting full adoption equilibrium is socially efficient.

Proposition 8 (Attaining Full Adoption) *There exists $\theta \geq 0$ such that if the frontrunnable user commits at $t = 1$ to make a payment θ to any validator who executes her transaction sent through the private pool, then (i) a full adoption equilibrium is attained, and the aggregate welfare is maximized; (ii) the expected payoff of all validators strictly increases; (iii) the expected payoff of the frontrunnable user does not decrease.*

In the partial adoption equilibrium, the total attack loss of the frontrunnable user, c, equals the total MEV extracted by validators and attackers. The portion of the MEV earned by validators, denoted as m, is in the form of fees paid by attackers, and the rest of the MEV, $c-m$, is captured by attackers. Frontrunnable users are willing to pay as much as c for the complete elimination of frontrunning risks, and validators are willing to fully adopt the private channel and eliminate frontrunning if the benefit of doing so is larger than m, i.e., the MEV they extracted in the partial adoption equilibrium. In this way, if the frontrunnable user pre-commits to make a payment $\theta > m$ to any validator executing her trade sent through the private pool, then it is incentive compatible for all validators to adopt the private pool. This maximizes aggregate welfare. Moreover, as long as $\theta < c$, the payoff of the frontrunnable user increases. This contract can be implemented in a straightforward manner. The relay service can set up a reward pool, that is, a smart contract which allows users to voluntarily deposit tokens into it. Any validator who joins the relay service and successfully mines a new block including transactions sent through this relay can claim the tokens deposited in the reward pool. Alternatively, private pools can also charge users flat fees for each transaction successfully executed. In other words, one should allow frontrunnable users to pay for their pre-trade privacy and incentivize validators to adopt private pools.

6 Empirical Analysis

In this section, we provide empirical support for the main implications of our model. Section 6.1 lists the model implications. Section 6.2 describes our dataset. Section 6.3 defines the key variables and stylized facts. Section 6.4 presents our empirical results.

6.1 Testable Implications

Our model generates the following implications:

1. A private pool will be partially adopted by validators (see Proposition 4).
2. Validators who adopt the private pool have a higher expected payoff than those who stay in the public pool. (See equation (1))
3. Users submit transactions through the private pool when the frontrunning risk is high (see Proposition 4).
4. Attackers' transaction fees increase with a private pool. This is implied from Proposition 6.

6.2 Data

We use transaction-level data from Uniswap and Sushiswap to identify frontrunning attacks. We run our own Ethereum node to get access to the blockchain history. A modified geth client is used to export all transaction receipts where a *swap* event was triggered by a smart contract of Uniswap or Sushiswap. Our dataset contains all swap transactions from block number 10000835 created on May 4, 2020 to block number 12344944 created on April 30, 2021. For the AMMs transactions in the data, we follow the heuristics described in [12] to identify frontrunning attacks and calculate their revenues.

We use the API services provided by Flashbots to collect transactions submitted through the private channel to the validators. We collect data starting from February 11, 2021, when the first Flashbots block was mined, till July 31, 2021. This choice eliminates the influence of the new fee mechanism introduced by EIP 1559 after August 2021.

We source the Ethereum block data from Blockchair. The data cover the period from May 1, 2020 to July 31, 2021. The data include the gas fee revenues earned by validators.

6.3 Definition of Variables and Stylized Facts

We describe the main variables used in our statistical analysis, and describe empirical regularities from our data set.

Adoption Rate of the private pool by validators. We estimate the adoption rate of the private pool in day t using the number of blocks mined in day t that contains Flashbots transactions divided by the total number of blocks mined in day t.

Validators' Revenue per Block. If a validator mines a block that contains transactions submitted through Flashbots, then his revenue accounts for Flashbots transactions in this block plus gas fee proceeds from transactions submitted through the mempool. If a validator mines a block that only contains transactions submitted through mempool, then his revenue consists of gas fees paid by those transactions. We do not account for the fixed block reward in our measure of validators' revenue.

Attackers' Cost-to-Revenue Ratio . For each frontrunning attack order identified, the attacker's cost-to-revenue ratio is measured by the gas fee paid by this attacker divided by the revenue of the frontrunning attack. Both the gas fee and attack revenue are in the unit of ether.

Users' Probability of Being Frontrun. For each transaction submitted through the public mempool, we examine whether it is frontrunnable and whether it has been frontrun using a methodology described in Appendix C.2. The probability of being frontrun in day t is the number of transactions that were frontrun in day t divided by the number of all frontrunnable transactions submitted on that day.

Proportion of Users' Transaction Sent Through the Private Pool. For each transaction submitted through Flashbots, we examine whether it would

be frontrunnable if were submitted through the public mempool. The proportion of transactions sent through Flashbots in day t is the number of frontrunnable transactions submitted through Flashbots during day t divided by the number of all frontrunnable transactions submitted during that day.

Descriptive Statistics and Stylized Facts.

The summary statistics of the data are shown in Table 3 in Appendix D. The estimated adoption rate of Flashbots is shown in Figure 2 in Appendix D. The average Flashbots' adoption rate by miner validators is about 35%, which supports our model implication that the private pool is at least partially adopted. For miner validators who join Flashbots, we plot the proportion of revenue extracted from Flashbots transactions in Figure 3 in Appendix D. It is readily observed that Flashbots transactions contribute a nontrivial (around 15%) portion to the revenues of miner validators who joined Flashbots. The distribution of the cost-to-revenue ratio of attackers is plotted in Figure 4 in Appendix D. A direct comparison of panel (a)-(c) indicates that the cost-to-revenue ratio for attackers who submit their transactions through Flashbots is higher than that of attackers who use the public mempool. The average cost-to-revenue ratio increased after the introduction of Flashbots. Figure 5 in Appendix D plots the daily average cost-to-revenue ratio of attackers in the public mempool and Flashbots. After the introduction of Flashbots, the cost-to-revenue ratio from transactions submitted through Flashbots steadily increases while the cost-to-revenue ratio from transactions sent through the public mempool decreased. Our model offers a plausible explanation for this observed pattern: as the validators' adoption rate of private channels increases, more attackers migrate from the public mempool to the private channel, which increases competition and raises transaction fees. Figure 6 in Appendix D plots users' probability of being frontrun (red) and the proportion of users' transactions submitted through Flashbots (black). The graph suggests that users migrate to Flashbots as they face higher frontrunning risk.

6.4 Empirical Results

We provide empirical support to the main testable implications of our model.

Miner Validators' Revenue. We estimate the following linear model to compare the revenues of miner validators who monitor the private pool against the revenues of miner validators who only monitor the public pool:

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

where t indexes the date, $MinerRevenue_t$ is the revenue of miner per block, γ_t is the day fixed effects, $\mathbb{1}_{Private}$ is a dummy variable for Flashbots blocks, and ϵ_t is an error term. We cluster our standard errors at the day level. The coefficient ρ_1 quantifies the change in revenue per block after a miner joins Flashbots.

The estimates in Table 4 in Appendix D indicate that miners who adopt Flashbots on average increases their revenues by about 0.16 ETH per block. This

Table 1: 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

finding supports our model implication that the expected payoff of validators who adopt the private pool is higher than the expected payoff of validators who stay only on the public pool. In addition, the coefficient estimates reveal that these relationships are statistically and economically significant.

Cost-to-Revenue Ratio of attackers We estimate the following linear models to compare the cost-to-revenue ratio of attackers before and after the introduction of Flashbots:

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

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

where $CostRevRatio$ is the cost-to-revenue ratio of attackers, $\mathbb{1}_{After}$ is a dummy variable which equals one after the introduction of Flashbots and zero otherwise. $\mathbb{1}_{Private}$ is a dummy variable which equals one for transactions submitted through Flashbots and zero for transactions sent through the public mempool. 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. The coefficient 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, after the introduction of Flashbots.

Table 1 (a) indicates that, after the introduction of Flashbots, the average cost-to-revenue ratio of attackers increased by around 0.09, an increment that is almost a third of the average cost-to-revenue ratio before the introduction of Flashbots (around 0.3). Table 1 (b) indicates that the average cost-to-revenue ratio of attackers who submitted transactions through Flashbots is 0.44 higher

than that of attackers who only submitted through the public mempool. These findings suggest that the increase in the cost-to-revenue ratio after the introduction of Flashbots can be mostly attributed to the use of Flashbots by attackers. All results are statistically and economically significant. The regression results support our model implication that the introduction of Flashbots increases the cost of attackers and lowers their welfare.

Migration of Users 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.

We find that an increase in the probability of being frontrun is positively correlated (60% correlation) with a higher proportion of transactions sent through Flashbots. Table 5 shows the regression details in Appendix D. 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, this regression supports our model implication that frontrunnable users migrate from the public to private pools when they face higher frontrun risk.

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

1. Capponi, A., Jia, R., Wang, Y.: Blockchain private pools and price discovery. In: AEA Papers and Proceedings. vol. 113, pp. 253–256. American Economic Association (2023)
2. Cong, L.W., Li, Y., Wang, N.: Token-Based Platform Finance. Journal of Financial Economics **144**(3), 972–991 (2022)
3. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 910–927 (2020)
4. Daian, P.: Mev for the next trillion, it's time to get serious... | flashbots (Oct 2022), https://writings.flashbots.net/mev-for-the-next-trillion
5. Gan, J.R., Tsoukalas, G., Netessine, S.: Initial coin offerings, speculation, and asset tokenization. Management Science **67**(2), 914–931 (2021)
6. Harvey, C.R., Ramachandran, A., Santoro, J.: DeFi and the Future of Finance. Working paper (2021)
7. Heimbach, L., Kiffer, L., Torres, C.F., Wattenhofer, R.: Ethereum's proposer-builder separation: Promises and realities. arXiv preprint arXiv:2305.19037 (2023)
8. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Unpublished manuscript (2008)
9. Park, A.: The Conceptual Flaws of Constant Product Automated Market Making. Working paper (2021)
10. Qin, K., Zhou, L., Gervais, A.: Quantifying blockchain extractable value: How dark is the forest? arXiv preprint arXiv:2101.05511 (2021)
11. Torres, C.F., Camino, R., et al.: 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). pp. 1343–1359 (2021)
12. Wang, Y., Zuest, P., Yao, Y., Lu, Z., Wattenhofer, R.: 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 (May 2022)
13. Weintraub, B., Torres, C.F., Nita-Rotaru, C., State, R.: A flash (bot) in the pan: measuring maximal extractable value in private pools. In: Proceedings of the 22nd ACM Internet Measurement Conference. pp. 458–471 (2022)
14. Yang, S., Zhang, F., Huang, K., Chen, X., Yang, Y., Zhu, F.: Sok: Mev countermeasures: Theory and practice (2022). https://doi.org/10.48550/ARXIV.2212.05111, https://arxiv.org/abs/2212.05111

A Example of Inefficiency in Blockspace Allocation

We provide an example to exemplify the two causes of blockspace allocation inefficiency. Suppose that the block capacity is 3, and there are four transactions whose values are $v_0 > v_1 > v_2 > v_3$ respectively. Let the first transaction be frontrunnable and the rest be non-frontrunnable. Consider the efficient allocation of blockspace that maximizes the aggregate welfare. Table 2a illustrates the efficient allocation where the most economically valuable three transactions v_0, v_1, v_2 are included in the block. The efficient allocation may not be achievable because of two types of inefficiency arise from frontrunning risk. Table 2b illustrates the type 1 inefficiency when frontrunning risk is severe ($c > v_0$). In such a case, the frontrunnable transaction (v_0) will not submit her transactions, and the blockspace is allocated to transactions with lower values, v_1, v_2, v_3. Table 2c illustrates the type 2 inefficiency where blockspace is taken by the frontrunning attack transaction. The attack transaction does not have value but merely transfer an amount c from the frontrunnable user, so the first two slots combined only add v_0 to the aggregate welfare. In such a case, only two slots are allocated to economically meaningful transactions, and the aggregate welfare is only $v_0 + v_1$.

(a) Efficient Allocation

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

(b) Type 1 Inefficiency

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

(c) Type 2 Inefficiency

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

Table 2: This table illustrates the two types of blockspace allocation inefficiencies. Table 2a illustrates the efficient allocation where aggregate welfare is maximized. Table 2b illustrates the type 1 inefficiency when frontrunning risk is severe ($c > v_0$). In such a case, the frontrunnable transaction will not submit her transactions, and the blockspace is allocated to transactions with lower valuations. Table 2c illustrates the type 2 inefficiency where the first block space is taken by the attack transaction. In such a case, only two slots are allocated to economically meaningful transactions.

B Technical Results and Proofs

Proof (Proofs of Proposition 1). 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	$(1-\alpha)(c - (f^{(B)} + \epsilon))$, $\alpha(c - (f^{(B)} + \epsilon))$	$\frac{1}{2}(c - (f^{(B)} + 2\epsilon))$, $\frac{1}{2}(c - (f^{(B)} + 2\epsilon))$	$\frac{1}{2}(1-\alpha)(c - (f^{(B)} + 2\epsilon))$, $\frac{1}{2}(c - (f^{(B)} + 2\epsilon))(1-\alpha) + \alpha(c - (f^{(B)}$
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 (Proof of Proposition 2). 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 (Proof of Proposition 3). 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 (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 2, and the attackers submit to both public and private pools as shown in Proposition 1.

Proof (Proof of Proposition 5).

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 (Proof of Proposition 6).

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.

Proof (Proof of Proposition 7).

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 (Proof of Proposition 8). 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.

C Empirical Methodology

C.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.

C.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

	N	Mean	SD	10th	50th	90th
Panel A: Miner validator Data						
Daily Private Pool Adoption Rate	171	0.343	0.239	0.01	0.346	0.613
Revenues of Validators on Flashbots (ETH)	377,366	0.972	17.82	0.235	0.606	2.2
Proportion of Revenue From Flashbots (ETH)	377,366	0.139	0.148	0.024	0.086	0.326
Revenues of validators on public mempool (ETH)	2,582,015	1.161	9.585	0.231	0.832	2.36
Panel B: attacker Data						
attack Revenue on Flashbots (ETH)	29,465	0.248	0.495	0.042	0.125	0.497
attack Cost on Flashbots (ETH)	29,465	0.182	0.363	0.032	0.092	0.371
Cost-to-revenue Ratio of attackers on Flashbots	29,465	0.755	0.151	0.51	0.801	0.901
attack Revenue on public mempool (ETH)	394,239	0.204	0.571	0.033	0.091	0.408
attack Cost on public mempool (ETH)	394,239	0.04	0.093	0.004	0.023	0.069
Cost-to-revenue Ratio of attackers which scan public mempool	394,239	0.309	0.239	0.021	0.261	0.662
Panel C: User Data						
Daily Probability of Being Attacked	80	0.165	0.034	0.120	0.165	0.209
Daily Ratio of frontrunnable transactions on Flashbots	80	0.033	0.038	0	0.01	0.09

Table 3: Summary statistics of the data set.

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.

D Empirical Results

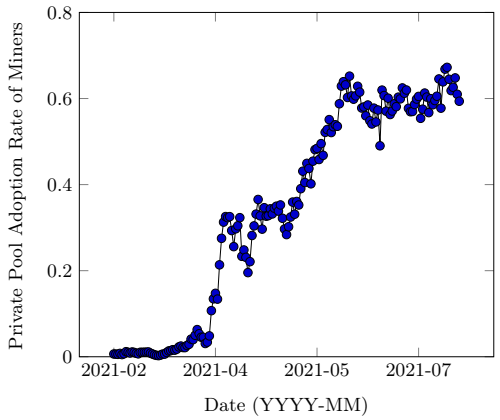

Fig. 2: Adoption rate of Flashbots.

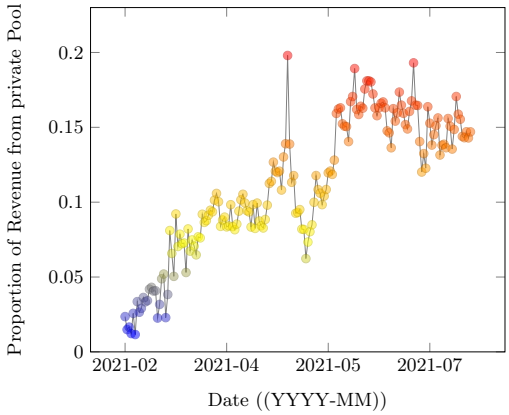

Fig. 3: Proportion of Flashbots miners' revenue coming from transactions on Flashbots.

Table 4: 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. Time fixed effects are included for all regressions. Standard errors are clustered at the day level. Asterisks denote significance levels (***=1%, **=5%, *=10%).

	Dependent variables: Miner 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

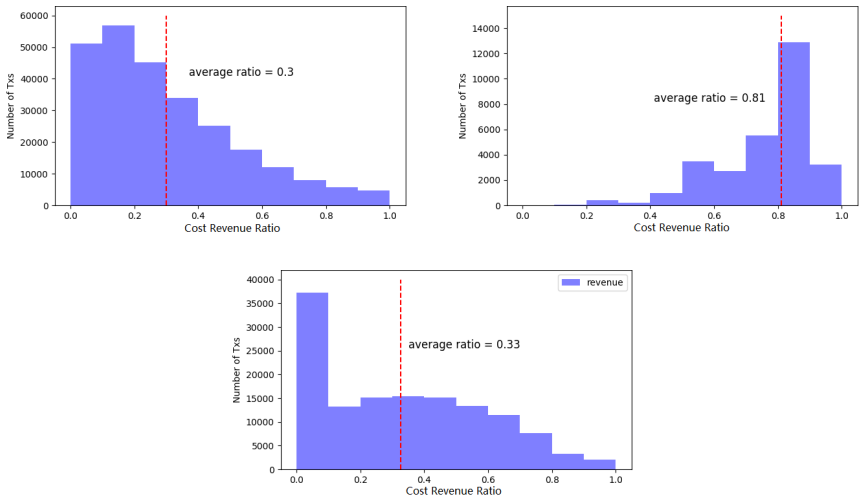

Fig. 4: Panel (a) - top left: Distribution of the cost-to-revenue ratio of attackers in the public mempool before the introduction of Flashbots. Panel (b) - top right: Distribution of the cost-to-revenue ratio of attackers on Flashbots. Panel (c) - bottom: Distribution of the cost-to-revenue ratio of attackers in the public mempool after the introduction of Flashbots.

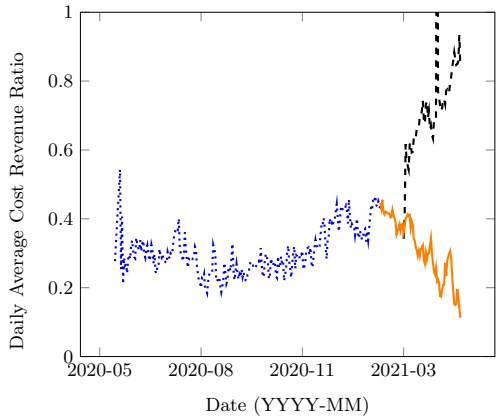

Fig. 5: Daily average cost-to-revenue ratio of attackers. Blue (dotted): attackers in public mempool before the introduction of Flashbots, Black (dashed): attack transactions through Flashbots, Orange (solid): attack transactions through the public mempool after the introduction of Flashbots.

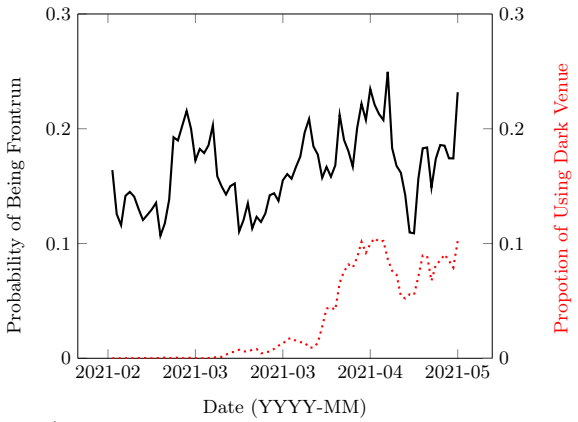

Fig. 6: Black (solid): the daily average probability that a frontrunnable transaction is attacked. Red (dotted): daily proportion of frontrunnable transactions submitted through Flashbots.

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