Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges

Philip Daian, *Cornell Tech*, phil@cs.cornell.edu
Steven Goldfeder, *Cornell Tech*, goldfeder@cornell.edu
Tyler Kell, *Cornell Tech*, sk3259@cornell.edu
Yunqi Li, *UIUC*, yunqil3@illinois.edu
Xueyuan Zhao, *CMU*, xyzhao@cmu.edu
Iddo Bentov, *Cornell Tech*, ib327@cornell.edu
Lorenz Breidenbach, *ETH Zürich*, lorenz.breidenbach@inf.ethz.ch
Ari Juels, *Cornell Tech*, juels@cornell.edu

***Abstract*—Blockchains, and specifically smart contracts, have promised to create fair and transparent trading ecosystems.**

Unfortunately, we show that this promise has not been met. We document and quantify the widespread and rising deployment of *arbitrage bots* in blockchain systems, specifically in *decentralized exchanges* (or "DEXes"). Like high-frequency traders on Wall Street, these bots exploit inefficiencies in DEXes, paying high transaction fees and optimizing network latency to frontrun, i.e.,[Tyler: I think you should get rid of the i.e. it's unnecessary - as opposed to the ie in the next paragraph which does serve to define the term priority ordering] anticipate and exploit, [Tyler: oxford comma? I think no should be no comma if you get rid of i.e.] ordinary users' DEX trades.

We study the breadth of DEX arbitrage bots in a subset of transactions that yield quantifiable revenue to these bots. We also study bots' profit-making strategies, with a focus on blockchain-specific elements. We observe bots engage in what we call *priority gas auctions* (PGAs), competitively bidding up transaction fees in order to obtain priority ordering, i.e., early block position and execution, for their transactions. PGAs present an interesting and complex new continuous-time, partial-information, game-theoretic model that we formalize and study. We release an interactive web portal, frontrun.me, to provide the community with real-time data on PGAs.

We additionally show that high fees paid for priority transaction ordering poses a systemic risk to *consensus-layer* security [Tyler: in blockchain systems]. We explain that such fees are just one form of a general phenomenon in DEXes and beyond—what we call *miner extractable value* (MEV)—that poses concrete, measurable, consensus-layer security risks. We show empirically that MEV poses a realistic threat to Ethereum today.

Our work highlights the large, complex risks created by transaction-ordering dependencies in smart contracts and the ways in which traditional forms of financial-market exploitation are adapting to and penetrating blockchain economies.

I. Introduction

Cryptocurrency exchanges today handle more than $10 billion in trade volume per day. The vast majority of this volume occurs in *centralized* exchanges, which hold custody of customer assets and settle trades. At best loosely regulated, centralized exchanges have experienced scandals ranging from high-profile thefts [28] to malfeasance such as price manipulation [17]. One popular alternative is what is called a *decentralized exchange* (or "DEXes").[1] In a DEX, a smart contract ([Tyler: a] program executing on a blockchain) or other form of peer-to-peer network executes exchange functionality.

At first glance, decentralized exchanges seem ideally designed. They appear to provide effective price discovery and fair trading, while doing away with the drawbacks of centralized exchanges. Trades are atomically executed by a smart contract and visible on the Ethereum blockchain, providing the appearance of transparency. Funds cannot be stolen by the exchange operator, because their custody and exchange logic is processed and guaranteed by the smart contract.

Despite their clear benefits, however, many DEXes come with a serious and fundamental weakness: on-chain, smart-contract-mediated trades are slow. (The average Ethereum block time is roughly 15s at the date of writing [15].) [Tyler: periods are weird here, i would have just have included it without period as a particle in the previous sentence] Traders thus [Tyler: Thus, traders] often attempt to take orders that have already been taken or cancelled but appear active [Tyler: comma,] due to latency. Worse still, adversaries can *frontrun* orders, observing them and placing [Tyler: their own] orders with higher fees to ensure they are mined first.

Past work has acknowledged "transaction ordering dependence" as an anti-pattern and vector for potential frontrunning [22, 25]. Unfortunately, these analyses have previously proved overly broad: virtually every smart contract can be said to have *some* potential dependence on transaction order, the majority of which is benign. As a result, effective practical mitigations for these issues have failed to materialize, and few deployed smart contracts feature ordering protections.

[1] "Decentralized" exchange is something of a misnomer, as many such systems have centralized components; most systems we call "decentralized" exchanges could [Tyler: should] more accurately be classified as non-custodial: users trade without surrendering control of their funds to a third party in the process.

Other work has focused on systematizing knowledge around smart contract frontrunning [14], including citing early public versions of this work, [Tyler: period. However, that work] but has not measured the size of this economy or formalized its connection to protocol attacks.

In this work, we explain that DEX design flaws threaten underlying blockchain security. We study a community of *arbitrage bots* that has arisen to exploit DEX flaws. We show that these bots exhibit many of the [Tyler: same] market-exploiting behaviors—frontrunning, aggressive latency optimization, etc.—common on Wall Street, as revealed in the popular Michael Lewis exposé *Flash Boys* [24]. We explore the DEX design flaws that spawned arbitrage bots, measure and model these bots' behavior, and illuminate systemic smart-contract ecosystem risks implied by our observations. Our main focuses are:

Pure revenue opportunities: A specific sub-category of DEX arbitrage representative of broader activity, these are blockchain transactions that issue multiple trades atomically through a smart contract and profit unconditionally in every traded asset. We choose these opportunities as a focus because their simplicity makes them especially amenable to study and measurement. We experimentally determine a lower bound on this economy of over USD 6M to date and describe its participating exchanges and bots.

Priority gas auctions (PGAs): Because pure revenue opportunities offer unconditional revenue, arbitrage bots often compete against each other by bidding up transaction fees (gas) in what we call PGAs. We formally model bot PGA behavior and observe a cooperative equilibrium. We show that empirical measurements of the evolution of bot PGA strategies validate key features of our model.

Miner-extractable value (MEV): We introduce the notion of MEV, value that is extractable by miners directly from smart contracts as cryptocurrency profits. One particular source of MEV is *ordering optimization (OO) fees*, which result from a miner's control of the ordering of transactions in a particular epoch. PGAs and pure revenue opportunities provide one source of OO fees. We show that MEV creates systemic consensus-layer vulnerabilities.

Fee-based forking attacks: We show that OO fees can incentivize miners to mount forking attacks. While fee-based attacks were previously studied theoretically in Bitcoin [11], we empirically demonstrate a current, realistic threat in Ethereum.

Time-bandit attacks: We show that high-MEV regimes in general lead to a new attack in which miners rewrite blockchain history to steal funds allocated by smart contracts in the past. We call these *time-bandit attacks*. Our experiments show that MEV from pure revenue profits and PGA bot fees suffice to enable time-bandit attacks *on today's Ethereum*.

Our results are surprising for two key reasons. [Tyler: I would throw some vspace below this sentence too, looks a bit lopsided otherwise]

First, they identify a concrete difference between the consensus-layer security model required for blockchain protocols securing simple payments and those securing smart contracts. In a payment system such as Bitcoin, all independent transactions in a block can be seen as executing atomically, making ordering generally unprofitable to manipulate. Our work shows that analyses of Bitcoin miner economics fail to extend to smart contract systems like Ethereum, and may even require modification once second-layer smart contract systems that depend on Bitcoin miners go live [23].

Second, our analysis of PGA games underscores that protocol details (such as miner selection criteria, P2P network composition, and more) can directly impact application-layer security and the fairness properties that smart contracts offer users. Smart contract security is often studied purely at the application layer, abstracting away low-level details like miner selection and P2P relayers' behavior in order to make analysis tractable (e.g. [8, 19, 39, 40]). Our work shows that serious blind spots result. Low-level protocol behaviors pose fundamental challenges to developing robust smart contracts that protect users against exploitation by profit-maximizing miners and P2P relayers that may game contracts to subsidize attacks.

To illuminate the behaviors explored in this paper, we release a web dashboard, frontrun.me, that presents PGA data in real time. We open-source all associated code and hundreds of gigabytes of raw data on the Ethereum PGA economy (with processed data).[2]

We hope our efforts in general offer insight into the broad, application- and consensus-layer risks created by order [Tyler: ordering] dependencies in smart contracts and into the effects of traditional financial-market exploitation on blockchain consensus.

II. Background

We now provide background required to understand PGAs.

A. Smart Contracts

Smart contracts are small computer programs executed without user intervention, often by a system that allows all of its participants to verify these programs' correct execution. Smart contracts often use a public blockchain network as the underlying infrastructure for their execution [34, 38].

Ethereum [38] is currently the largest smart contract system that is Turing-complete, i.e., allows encoding of arbitrary smart-contract functionality. Ethereum smart contracts have been used or proposed for a range of complex transaction types, including shareholder voting [27], stakeholder-owned investment funds and vehicles [9, 13], fair exchange protocols for goods [40], complex key management solutions [39], video games [20], virtual casinos [29], and more.

The most popular smart contracts on Ethereum by daily active users primarily concern virtual sub-currencies called *tokens*. These tokens can represent any scarce item, e.g.

[2]Our Github repository at https://github.com/initc3org/flashboys2 contains all infrastructure, data processing, and visualization code and data, plus code for our original arbitrage trade bot.

collectible resources in a video game [20] or shares in a venture [16]. The latter fueled a 12 billion USD token-based capital-investment craze called the "ICO boom". DEXes are a popular type of smart contract that allow users to trade such tokens in a non-custodial manner [37].

B. Gas and Fees in Ethereum

An Ethereum transaction either sends money to a non-executing *account address* or sends input data (and possibly money) to a *smart contract address*, representing a program stored in the network's state. Transactions are gossiped to all of the nodes in the underlying peer-to-peer network to signal their availability for inclusion in a future mined block. Transactions can be in one of three states: *unconfirmed* and not yet mined, *confirmed* and considered to have been executed, or *rejected* as invalid by the network of Ethereum peers.

Ethereum transactions consume *gas*, a pseudo-currency reflecting the number of computational steps performed by a miner (and other network nodes) executing a transaction. Ethereum contracts may contain complex logic, loops, etc., so their gas consumption can only be determined via execution. Ethereum meters gas consumption via a fixed mapping from contract op-codes to units of gas [38].

Every transaction submitted to the network for mining specifies a *gas price*, the per-gas-unit rate the sender will pay in Ether (ETH). The gas price times [Tyler: multiplied by] the units of gas consumed determines the *fee* in ETH paid by the transaction sender to the node that ultimately mines the transaction. (Bitcoin fees, by contrast, depend simply on transaction byte lengths.)

Transactions must also specify a *gas limit*, the maximum number of steps a network node should attempt before rejecting a transaction. The gas limit prevents infinite loops and other DoS vectors, and allows immediate verification that a sender has adequate available funds to pay the transaction fee—up to gasPrice · gasLimit ETH.

Clients can also perform what is called a "gas replacement" transaction, resubmiting a transaction with a higher transaction fee (gasPrice · gasLimit) in the hope that miners will more quickly incorporate the transaction into a block. The mechanism for doing this is nonce-based. Each transaction issued on the network carries a *nonce*, and valid canonical blockchains must only include one transaction per (account, nonce) pair on the network. When a user reissues an unconfirmed transaction with the same (account, nonce) pair but a higher gas price, a miner will prefer the reissued transaction—with its ostensibly higher transaction fee—to the replaced one.

Note that a higher gas price only corresponds to a higher fee if the reissued transaction *uses the same amount of gas as the replaced one*, which may not hold true. Mining software assumes this to be the case, in part because computing gas consumption incurs the computational burden of executing a contract. Arbitrageurs leverage this heuristic to pay reduced transaction fees, as we detail in Section V-A.

C. Continuous-Limit Exchanges

Classic exchanges for trading stocks, commodities, and even cryptocurrencies generally share an accepted and common exchange design known as a *continuous-limit orderbook*. Such an orderbook consists of a list of all open offers from buyers and sellers in the system. Prospective buyers place a *limit buy* order, which specifies a maximum price at which they are willing to buy an asset; sellers correspondingly place *limit sell* orders. A centralized counterparty, the *exchange operator*, matches buyers and sellers, completing transactions automatically when there is a sell order on the books with a lower price than a buy order on the books. Orders are matched and / or placed on the books continuously by the exchange operator, which processes orders as quickly as possible in the order they are received. As soon as orders match, they are processed by the operator and trigger balance changes.

D. Decentralized Exchanges (DEXes)

DEXes manage continuous-limit order books using smart contracts. Traders / users hold their assets on chain and the smart contract plays the traditional role of exchange operator.

Order books are typically maintained off-chain. In some DEX designs, a counterparty selects a fresh order in the order book and presents it to the smart contract with a signed counterorder. The smart contract executes the order and counterorder, clearing the order from the order book. Traders themselves thus perform order matching. This approach is used by Etherdelta and some applications of the 0x protocol. [Tyler: In an] An alternative approach, used by, e.g., IDex and Paradex, the exchange itself performs matching off chain and submits order / counterorder pairs to the smart contract for processing.

A more radical DEX design, called an *automated market maker* [3], bypasses order books altogether. The DEX consists of a smart contract that itself holds a reserve of tokens and/or Ether. Consider two assets A and B. The contract allows a user to trade between A and B at any time using its reserves as a counterparty, at a set rate. If a user buys A using B, the price of A denominated in B offered by the smart contract for the *next trade* is increased. If a user instead sells A for B, the price decreases. In this way, single parties can trade without counterparty discovery or matching. Consequently, constant arbitrage between these and other exchanges is required to keep the rate offered in lockstep with the market rate for a commodity. Uniswap [35] and Bancor [1] are examples of such exchanges.

E. Frontrunning and Profits through Arbitrage

Traditional exchanges experience a classic form of predatory market behavior called frontrunning [4].[3] In regulated markets, frontrunning is often illegal, and has resulted in prosecutions [26] and tarnished the reputations of financial institutions caught practicing it.

Frontrunning generally exploits information asymmetries created by power structures within a financial structure, e.g., brokers having privileged access to user information. Because there is no single party playing the role of a broker in decentralized systems, information asymmetries can arise for actors in advantageous positions in underlying infrastructure.

Frontrunning can also occur based on changes in public market information (for example, reacting to breaking news that impacts stock prices). In this form, it is not only legal, but serves as the basis of a multi-billion-dollar high-frequency trading economy. One potential source of profit is price discrepancies across exchanges trading the same or correlated assets. Another is information asymmetries in the speed of processing or interpreting news.

Automated market bots from high frequency trading firms compete to profit from both at extremely high speed. They regularly build physical networks costing billions of dollars and approaching speed-of-light transmission across considerable distances. Many economists view this behavior as a zero-sum game that profits exchanges in the long term, and argue formally that the existence of such rents is a fundamental limitation of market design that is a natural consequence of arbitrage opportunities across exchanges [7]. This, however, remains a controversial viewpoint.

We explore both cases where bots frontrun user orders and cancellations directly, e.g., in the event of a typo or market structure weakness, where bots exploit market inefficiencies to extract rent. We argue that both degrade the economic security of the underlying consensus protocol.

III. From Decentralization to Arbitrage

In this section, we take a deep dive into a particular example of frontrunning, arbitrage, and high-frequency automated trading on a decentralized exchange. This concrete example will provide context for the remainder of our discussions on modeling this market and of its impact on the security of the underlying smart contract systems.

One source of potential profit, [Tyler: that is -] price differences, seems inherent in an environment such as smart-contract-based exchange. Today, blockchains operate with transactions processed in discrete batches (blocks). Furthermore, transactions are inherently dependent and therefore serial: order failures depend on past order attempts, and in some exchanges prices depend directly on order history. With multiple exchanges operating on the same system, it is possible that price differences will occur across exchanges while transactions in a block [Tyler: comma,] and therefore trades on exchanges [Tyler: comma,] are executed sequentially.

A. Smart-Contract-Enabled Trade Atomicity

Smart contract arbitrage opportunities have an additional, distinctive characteristic absent in traditional cross-exchange arbitrage. Because of the atomic batch-based processing of transactions, and because transactions can themselves be initiated by smart contracts, is possible to build bots that trade across exchanges through *proxy contracts*. These proxy contracts can execute *batches* of orders sequentially within a single transaction, reverting previous trades by throwing an exception if any trade in the batch fails.

This means arbitrageurs have the opportunity to compose single transactions that execute multiple trades across multiple exchanges atomically, with an all-or-nothing failure model. One example of such a transaction is buying an asset for price x and selling it immediately for price $x' > x$; if performed atomically, these transactions together generate guaranteed revenue in the base asset. For example, a smart contract proxy could execute a trade buying a type-X token for 2 ETH, and another selling it for 3 ETH. If both orders are on the books on some decentralized exchange, a smart contract executing both guarantees a revenue to the arbitrageur of 1 ETH.

In traditional cross-exchange arbitrage, trades are viewed probabilistically, as there is a high likelihood that one of two trades will succeed while the other fails. This makes smart-contract-based arbitrage in many ways simpler to observe, analyze, and study than traditional cross-exchange arbitrage, as bot intent is often explicit in order requests.

In our measurements, we focus on a small subset of these multi-trade arbitrage opportunities, which potentially involve multiple decentralized exchanges. We restrict our focus to *pure revenue opportunities*. In these opportunities, a single smart-contract based transaction executes multiple trades across one or more exchanges, and the transaction generates revenue for the trader in every traded asset. A range of more complex and nondeterministic bot behaviors exist, described in Appendix A, but they are outside the scope of this work.

B. Pure Revenue by Example

Figure 1 shows one example of a pure revenue transaction, executed on November 15, 2018. In this transaction, two trades are executed on a decentralized exchange, TokenStore, which features a design conceptually similar to that of Etherdelta. The first executed trade buys Free Coin, an obscure token.[4] By inspection, this difference in rates is a clear result of someone using the exchange API and committing an off-by-one error, offering to buy tokens at 10x the market rate. This created a cross in the order book (sell order at more than a buy

[3] Frontrunning derives its name from the days of physical orderbooks. Brokers who would trade against information they gained from watching other participants' behaviors. They would do so by literally running in front of competitors about to place orders. [Tyler: you need to find a citation for this because I spent a long time looking for this quote and ultimately could not find the original quote - the wikipedia page cites Edwin Lefèvre (1923). Reminiscences of a Stock Operator. but if you actually pull the epub for that book and search for "running" in the book you cannot find the word running describing frontrunning. I tried it.]

[4] As of writing on Mar 13 2019, Free Coin is listed at currency rank 303 by market cap https://coinmarketcap.com/currencies/free-coin/.

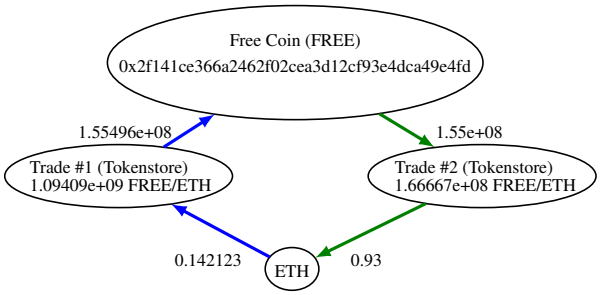


Fig. 1. Example pure revenue opportunity observed in Ethereum transaction 0xc889bd13594f75e4dd824f04f0c2ad03896cb7ec6518df02455e9560367bb9c4, exploiting an orderbook cross on the TokenStore DEX. Edges of the same color are generated by a single trade. As a pure revenue opportunity, it generates net profit in both FREE and ETH.

order would pay), which when both executed by the same arbitrage counterparty, generates the flow of funds in Figure 1. While this opportunity probably arose from a typo, a variety of revenue sources exist. For example, inconsistent price feeds, or variance across exchange designs that respond to market movements at different speeds can also create pure revenue.

Note that both orders are executed inside a single Ethereum contract, and are executed in an atomic batch through a smart contract proxy. In general, we model each opportunity as a graph as shown in Figure 1. Edges represent currency flows, and nodes are either exchanges nodes (which receive one currency and output another at the stated exchange rate), or asset nodes (which are either sources or sinks for same-color trade subgraphs, depending on whether they are bought or sold). We label exchange nodes with the exchange name and trade rate, and asset nodes with the asset symbol.

The net revenue in each traded asset is the sum of inflows minus outflows in the corresponding asset node. In the trade in Figure 1, the revenue for ETH was therefore $0.93\text{ETH} - 0.14123\text{ETH} \approx .79\text{ETH}$, or approximately 267 USD in equivalent value (at November 15, 2018 prices of 338.15 USD per ETH). Approximately 496,000 residual FREE was also left in the arbitrageur's account, although less significant and liquid than the ETH revenue.

To calculate the transaction profit, we subtract the cost from the revenue, in this case the gas paid by the transaction that was mined for the arbitrageur. Transaction 0xc889...b9c4 paid a gas price of 134.02 Gwei (the canonical unit for representing gas rates; 1 Gwei = 10^{-9} ETH. The transaction used $113,265$ gas, approximately equivalent to $113,265$ computational steps.

It is worth noting that transactions that cancel their bids can use less gas, as they pay only for attempted execution (execution requires costly computational steps). This will be relevant to later models in Section V-A. Furthermore, complex conditional transactions, price queries, or other computationally intensive order preferences will obviously cost more, making optimizing for gas consumption important in the development of competitive bots. The total cost of this transaction was therefore $113265 \cdot 134.02$ Gwei $= 0.01518$ ETH, or around USD 5.13 at the time of the transaction. The associated profit was therefore $\approx .79 - .01518 = .77$ETH, or USD 267.

C. PGAs, Ordering Fees, and... HFT?

Ethereum transactions are routed in a peer-to-peer gossip multicast protocol by client node software. This means that all transaction information is available to all participants in this network, but earlier to participants with advantageous positions in the gossip topology. Additionally, nodes can simulate the outcome of every transaction given the current or expected system state. Once an arbitrage transaction is submitted, therefore, the sequence of trades it involves is publicly known by the network's peer-to-peer nodes.

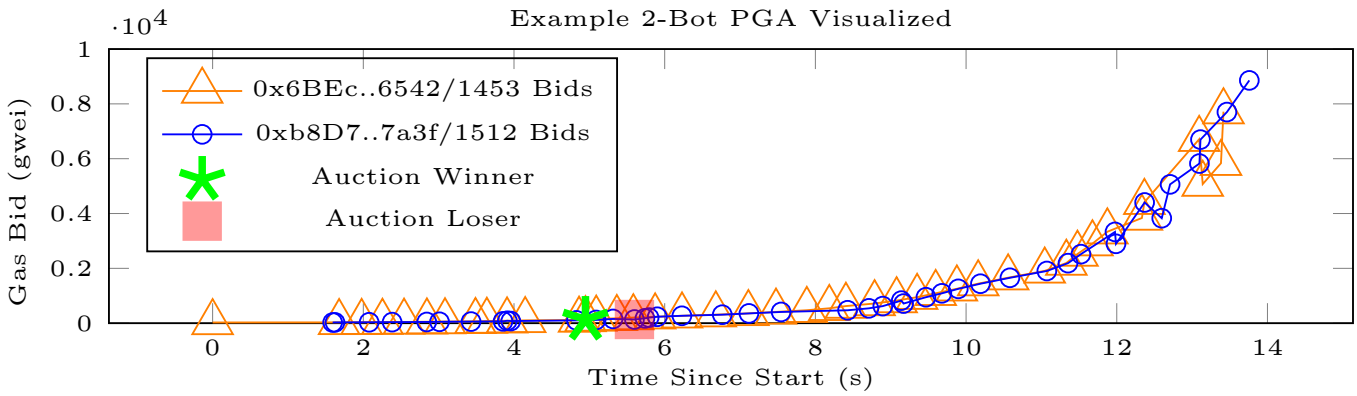
A natural question then arises: how is priority determined between arbitrageurs? Because each pure profit opportunity carries some computable profit p and is broadcast globally, a competitive game naturally ensues among arbitrage bots to be the first to execute an atomic transaction that exploits the opportunity. The mechanics of the system dictate that all subsequent transactions in the game will fail. How this particular game plays out depends on the peer-to-peer relay network mechanics of the underlying blockchain, as well as mining pool strategies and order book designs in the underlying exchanges. We present a greatly simplified model of this game formally in Section V-A.

In Ethereum, the PGA game we observe consists of transactions issued with the same (account, nonce) pair, expressing a bid to a miner, where the miner is paid gas as described in Section II-B. We call this interaction of issuing repeated bids a *priority gas auction*, or PGA.

To place bids on how much they will pay for the privilege of the arbitrage opportunity, an arbitrageur simply issues an atomic bundle of trades through a smart contract transaction they wish to execute, along with some gas price g. If they later wish to increase this bid, either for strategic reasons or because they notice a higher bid that would eliminate their profit issued on the peer to peer relay network, they simply reissue the transaction with the same nonce and a higher gas price g'. If two bots are bidding against each other, what emerges is essentially an auction, as shown in Figure 2. It is in a miner's interest to order first whichever transaction offers the highest price (if the miner does not plan on itself arbitraging the market). This approach will encourage arbitrage bots to bid each other up for transaction priority.

[Ari: I proofread up to this point.]

Figure 2 shows the action that occurs in such an auction from the point of view of the network. Each row in the associated table is a transaction observed by our Ethereum monitor on the peer to peer network, using the experimental harness described in Section IV. In this example, we see two accounts, 0x6BEc...6542 with nonce 1453 and 0xb8D7...7a3f with nonce 1512, bidding against each other for priority. These accounts issue transactions with ever increasing gas prices: bot 0x6BEc...6542 issues 42 transactions in 13.4 seconds, and 0xb8D7...7a3f issues 43 transactions in 12.1 seconds. In 4.94 seconds, the auction is over. The transaction that eventually ends up mined with priority is shown with a green X, and is



Seconds Elapsed	Quantity @ Price Bid	Ethereum Transaction Origin (Public Key Hash)	Nonce	Transaction Hash
0.000	192085 @ 25.10	0x6BEcAb24Ed88Ec13D0A18f20e7dC5E4d5b146542	1453	0xd32653ca9694a6d1299335f3c04f74cc159bee48c1d32d3a421db08c638ffc78
1.593	231520 @ 25.00	0xb8D76f4BC2518F8eb508bf0Ccca76f8F9DD57a3f	1512	0xb901e6dc2c229fd9105448fcc23eaebdedb476c21b6c6e7ddf8d2df4e838d2c7
1.624	231520 @ 28.75	0xb8D76f4BC2518F8eb508bf0Ccca76f8F9DD57a3f	1512	0x9f592504eb71a7452b7a395a7f5ecd34eaa5d090da1162e74221562af54c8f67
1.679	227534 @ 28.81	0x6BEcAb24Ed88Ec13D0A18f20e7dC5E4d5b146542	1453	0x83e2a6774654a9540c3fad8837afcc88b4c932ab2374819254f887305c3a4b22
...	...	...	...	...
4.949	227534 @ 134.02	0x6BEcAb24Ed88Ec13D0A18f20e7dC5E4d5b146542	1453	0xc889bd13594f75e4dd824f04f0c2ad03896cb7ec6518df02455e9560367bb9c4
5.599	231520 @ 133.76	0xb8D76f4BC2518F8eb508bf0Ccca76f8F9DD57a3f	1512	0xaa86d782328c0c9c422e3f2a3170ff41ae21a27ad395c48db76b0080898f85db
...	...	...	...	...
13.383	227534 @ 5834.77	0x6BEcAb24Ed88Ec13D0A18f20e7dC5E4d5b146542	1453	0xb0dc97140394c5f65332ebc459d5e66f89099dbb4d335c866b32280270102858
13.416	227534 @ 7716.48	0x6BEcAb24Ed88Ec13D0A18f20e7dC5E4d5b146542	1453	0x1825be6951577e72a1dafc8de564ce1ccfe5d284173e11e77b2e7f6b1b44571c
13.462	231520 @ 7701.08	0xb8D76f4BC2518F8eb508bf0Ccca76f8F9DD57a3f	1512	0xa9823358c99149f0e6343c604c35988468d01d02868437d8251b3cee282dc92b
m13.759	231520 @ 8856.24	0xb8D76f4BC2518F8eb508bf0Ccca76f8F9DD57a3f	1512	0x366c30a534b5f3d8a6d251f97d401997624d1fe8d3af07ede4d19105dc970942

Fig. 2. One example PGA that was observed over the Ethereum peer-to-peer network, resulting from the profit opportunity in Figure 1. The top graph shows the gas bids of two observed bots over time, while the bottom table details the first and last two bids placed by each bot and the two mined bids (center).

transaction 0xc889...b9c4 issued by bot 0x6BEc...6542. This bot is considered the *winner*, and pays the full gas price to reap the revenue as described in the previous section. The transaction was mined in block 6709727 by the mining pool "MiningPoolHub". The transaction shown in the red X, with hash 0xaa86...85db by bot 0xb8D7...7a3f, is also mined and included in the final block, as each (account, nonce) pair can include one transaction in the next block to be mined as per the protocol. Miners should also always include even failed transactions, as these transactions pay for attempted execution. Note that while the winning transaction used 113,265 gas in execution, the losing transaction paid for 33,547 gas units, a far smaller sum. In game theoretic terms, each auction represents a variant of *all pay* auction, where instead of paying their full bid, the loser is forced to pay a percentage to the miner.

IV. Arbitrage Prevalence Measurement Study

We now present empirical measurements of the prevalence of arbitrage. We discuss our methodology (and its limitations).

A. Experimental Setup

We first describe the experimental harness used to observe PGA transactions, displayed in Figure 3. On-chain data is not sufficient to analyze PGAs, as all but the final "winner" transaction are discarded by the network. Furthermore, because transactions are replaced so quickly and nodes don't propagate replaced transactions, many transactions are never even propagated to all Ethereum nodes.

No tools existed for analysis of unconfirmed and rejected transactions, so we wrote our own. We forked the Go-

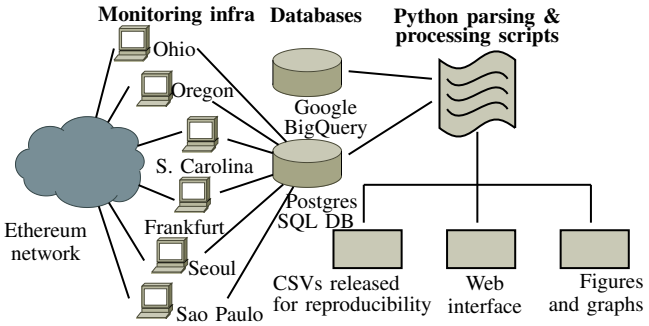


Fig. 3. Deployed PGA measurement infrastructure architecture.

Ethereum client to record unconfirmed transaction in the mempool. We deployed six [Tyler: phil, changed by me for accuracy, as discussed] geodistributed nodes across multiple data centers, with timestamps synchronized to the nanosecond level by NTP. We collected an observation every time one of the 256 nodes peered with one of our deployed modified nodes relayed a transaction to us. We collected nine months of data, amounting to over 300 gigabytes, including 708,385,840 unique observations of PGA arbitrage bots. Node locations are shown in Figure 3, although not all nodes were online throughout the experiment. (We ultimately sacrificed timing resolution to reduce prohibitive ongoing costs.)

Because it is technically infeasible to store observations of every Ethereum transaction, we focus on a list of suspected arbitrage bot transactions. This list is seeded with accounts we observe performing pure revenue transactions on the

blockchain, and is updated dynamically any time a high-value gas replacement transaction is seen at an order of magnitude over current gas market price. We also built a web interface to manage the uptime of our nodes and associated bot lists.

We supplemented this mempool data we collected on PGAs with on-chain data sourced from Google's BigQuery Ethereum service (as well as other on-chain metadata, like block timestamps), allowing us to parse logs for successful transactions and determine their profit. We also used daily price data from coinmetrics.io for USD conversions.

We then developed a suite of Python scripts to combine and analyze this data. These scripts use a heuristic to place all observations on a timeline, identifying a PGA whenever a high-value gas replacement transaction occurs. All transactions in a time interval around this observation are then considered part of the "auction," and broken down per bot. The scripts also aggregate [Tyler: aggregates?] meta-statistics on PGAs, calculating strategy and latency trends in observed bots. This data is used to generate both our web interface and the figures in this paper. We release all source and processed/derived data in CSV format for further analysis by the community on our Github. [Tyler: footnote: link to github url?]

1) Instrumentation Limitations: The above instrumentation has limitations that may affect our data quality. We describe them at a high level here, and mention any impact these limitations may have on our derived results throughout.

It is possible that a transaction will be replaced before reaching our nodes, though this seems unlikely, since each of our observed transaction tends to have hundreds of associated observations. More critically, our time-slicing method for identifying PGAs may lump together unrelated bot activity into a single "auction." Manual inspection suggests that because auctions are relatively infrequent (every few hours), the majority consist of correlated bot activity. Time-slicing could also harvest unrelated transactions from other arbitrage bots that are not PGA behavior; we prune these in our aggregate statistics by only including bots that have placed at least 4 "bids" in an observed PGA, and as a result may lack data on bots that place < 4 bids. We also may miss bots whose addresses are not in our PGA lists, leading to missing bidders in certain auctions.

Lastly, our instrumentation calculates pure revenue opportunities by parsing transaction logs on supported exchanges for transactions that contain more than two trades executed by a smart contract. We support only a limited subset of popular DEXes, omitting revenue opportunities on unsupported exchanges. Because we aim to establish a lower bound and thus conservative results in this work, we feel this is acceptable. Our supported exchanges include the top five DEXes by sustained volume at the time of infrastructure development.

This subset still proves sufficient to provide substantial insight on the PGA market not afforded to regular nodes in a blockchain context, highlighting the limits of the "transparency" afforded users by these systems.

B. Observations

We now describe key results of our data gathering.

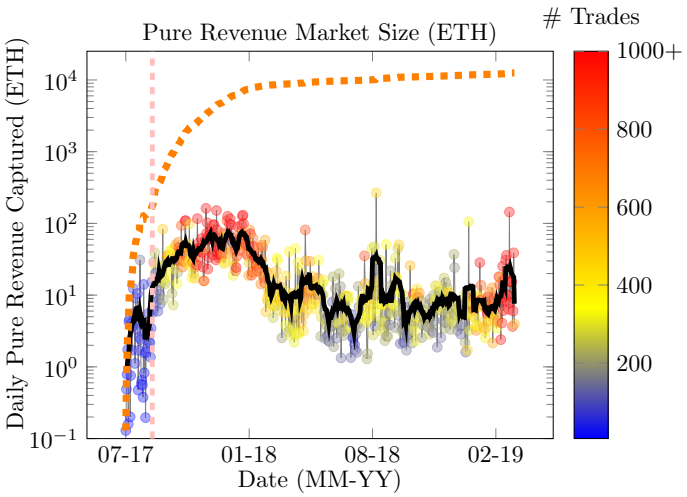


Fig. 4. Lower bound size of the observed pure revenue market, ETH. Black line shows a 14-day moving average of daily price, while the dotted orange line plots cumulative market size. Scatter plot colorings/shading indicates number of daily pure revenue trades. The pink vertical line shows the release of [2] associated with early versions of this work becoming public.

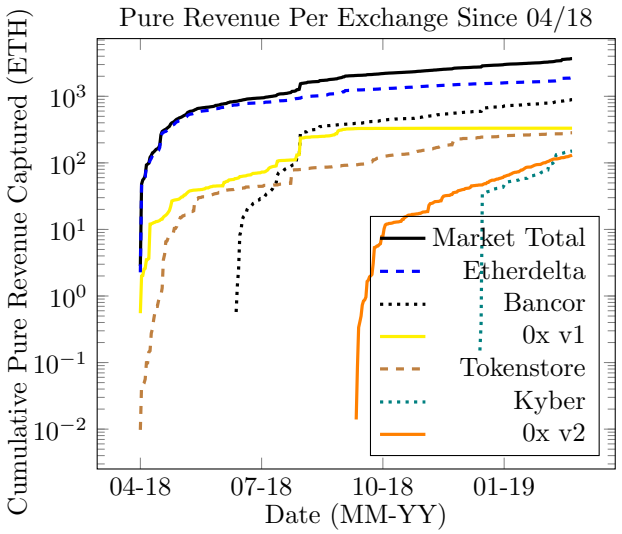


Fig. 5. Top exchanges by cumulative pure revenue offered bots since 04/18.

Figure 4 shows the size (breadth) of the arbitrage market, denominated in ETH. We choose to use ETH denominations in the rest of the work as they display a relatively consistent distribution, as shown in the figure. USD revenues are more volatile, as is visible in the market revenue graph. We discuss this in Appendix B-A.

This distribution is consistent with common understanding of arbitrage as sourced from market structure design, as ETH is the most liquid traded token on this market. As seen in this graph, after the initial market development in late 2017, a relatively active period of arbitrage occurred during which

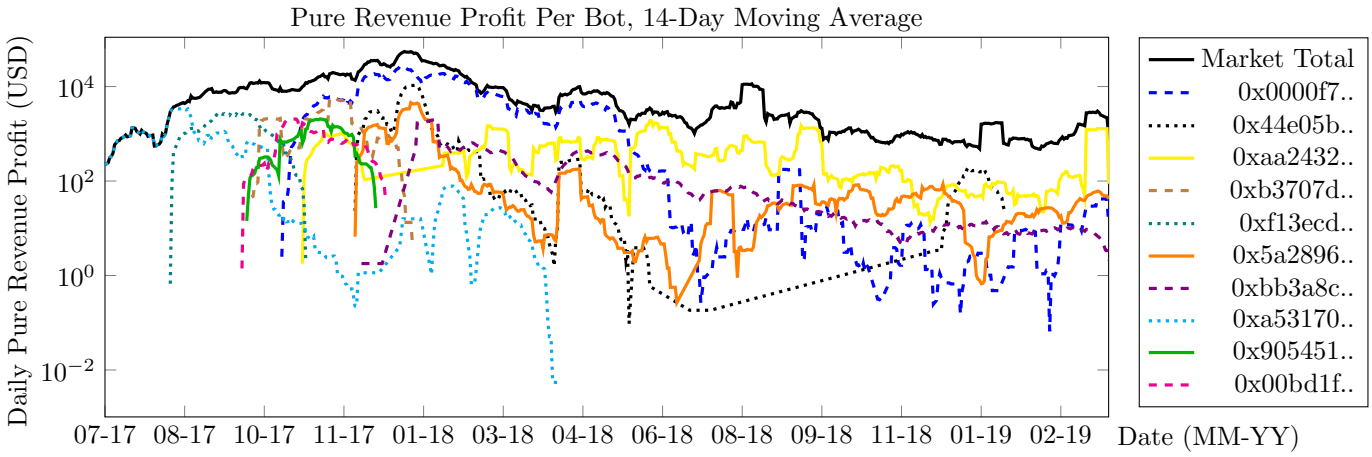


Fig. 6. Profit (revenue minus estimated costs) for pure revenue bots over time. We observe that the median profit is 65% of the size of the pure revenue opportunity across 138,948 observed pure revenue transactions.

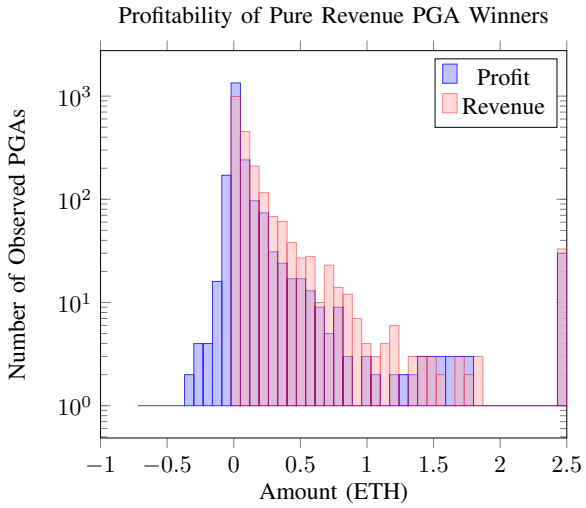


Fig. 7. Distribution of profit and revenue in competitive PGAs. We focus on amounts from -1 to 2.5ETH, which by inspection comprise the majority of the distribution body. 29 of 2,135 profits fall outside this range.

bots performed often 1,000+ daily trades for a daily revenue of 10-100 ETH. Later, the market matured into a more steady and consistent profit distribution, with 1-10ETH available for arbitrage daily. Recently, an increasing number of pure revenue trades is observed, showing the trend of the DEX market to smaller retail transactions and more efficient market designs, decreasing the average opportunity size but presenting more frequent opportunities to a more efficient bot market. The limitations in our instrumentation mean, however, result in an underestimate of market size, as described in Section IV-A1.

Interestingly, Figure 4 also highlights the origin story of this work. In August 2017, we published a preliminary report on the dangers of decentralized arbitrage and associated bot designs in [2]. The date of our blog release is shown as the vertical line on the graph. As part of this post, we actively probed the Etherdelta orderbook for pure revenue opportunities, measuring available profit at 4,472.75 USD / day or 1.6M+ USD / year. We executed our own trading bot to confirm the technical feasibility of profit, observing a total 58% rate of success in capturing pure revenue for our own bots (soon outcompeted by other bots after the release of our post). We calculated the expected daily profit of an arbitrage bot at the time, before most decentralized exchanges were online, at 0.32 ETH/day $\cdot 58.3\%$ success $\cdot 45 - 0.004 \cdot (1 - 58.3\%)$ failure cost * 45 observed opportunities = 8.32 ETH (2500 USD) daily profit. Note in Figure 6 that the majority of profitable bots today joined shortly after our public release, which inadvertently sparked a thriving cottage bot economy!

Figure 5 shows the breakdown of the pure revenue market by exchanges, since 04/18 (the first significant pure revenue opportunities outside Etherdelta, representing new exchanges coming online). An oligopolistic pure revenue market is observed, with Etherdelta generating the majority of observed pure revenue. Nonetheless, a range of other exchanges furnishes a growing and relatively consistent distribution of opportunities for bots. Because we only support certain exchanges, unsupported exchanges may be missing from this graph, and the market total provides a lower bound only.

Figure 6 shows the top ten bots we observe in the competitive pure revenue market, and their associated profits. The cost we measure for these bots includes only the gas fees for the pure revenue transactions they make, and thus may underestimate their total fees (e.g. for failed transactions). We also do not include any bots that do not trade on our supported exchanges.

As in the exchange breakdown, this figure suggests an oligopolistic market, with single bots often dominating the profit space for extended time periods (for example, 0xaa24... dominates the recent pure revenue space, where 0x0000... [Tyler: is this address right? lot of leading 0s] experienced a long period of dominance through late 2017 and early 2018). This diagram also shows the behavior of top bots exiting the market after failing to update their strategies, for example 0xa53... exiting around 03/18.

Interestingly, the revenue and profit graphs for these bots have nearly identical shapes (the revenue graph is described

in Appendix B-A). Profit-to-revenue ratios are relatively well distributed, with a median of 65% of the revenue of an opportunity captured by the winning bot according to our profit heuristics. Most opportunities are also uncontested, providing the bots a steady stream of income.

Thusfar, we have focused exclusively on pure revenue opportunities, which may lead to PGAs or may be claimed by a single bot uncontested. A natural question becomes whether the profitability of opporutnities persists in a competitive market, or whether competition among bots creates a generally unprofitable zero-sum or negative-sum environment.

Figure 7 describes the breakdown between the profit and revenue of observed pure revenue opportunities involved in priority gas auctions, indicating that at least one bot placed a competitive gas replacement bid to capture such an opportunity. While there is a spike around 0 profit, indicating that many PGA opportunities *are* zero or negative sum for their players, the vast majority of these opportunities see relatively insignificant costs, and the profit distribution still provides a mean profit to winning bots that offers them a majority of associated revenue. This confirms our suspicious that bots may be engaging in uncoordinated cooperation to maintain the profitability of the PGA market at the expense of miners, which we model fully in Section V.

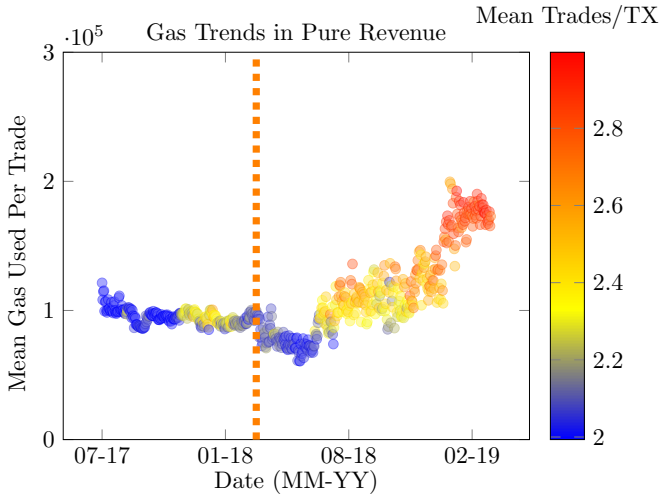


Fig. 8. Trends in quantity of gas used per trade by observed pure revenue transactions. The vertical line indicates the release date of a gas optimization token by the authors and others as part of this study, described in [6].

A final natural question to ask about the market is the extent to which the gas used in a transaction (representing the "quantity bid" in a PGA) is an important competitive optimization vector for pure revenue bots. Figure 8 shows this optimization over time. Before 04/18, primarily Etherdelta trends are observed. As we expect, we see bots optimizing their gas costs with a clear downward trend over time, despite relatively consistent execution of primarily pure revenue transactions containing two trades per transaction.

The advent of more sophisticated exchanges shows market maturation, with more complex opportunities executed that require more gas per trade. The number of trades per transaction has also increased, now averaging well over two trades per transaction. Part of this trend is due to "liquidity pooling" exchange designs like Kyber, which allow use of an exchange contract as a proxy to trade on other exchanges.

The dotted orange line on this graph shows the authors' public release of a token called GasToken [6]. It leverages a feature in Ethereum's incentive model enabling arbitrageurs to perform *gas arbitrage* over time, banking gas at below market rates and deploying it to win PGAs. The mechanism by which this gas arbitrage is performed yields a transaction refund in the quantity bid to miners, tricking miners into accepting smaller than expected bids. Because this allows bots to bid higher gas prices for less quantity at the same level of cost, this token is now a requirement for participating competitively in the pure revenue arbitrage market. This is reflected in our trends graph as a sharp downtrend in gas quantity bots demand from miners after release of our token. This anecdote provides unusual insight on consequences of performing active experiments on emerging competitive markets.

V. Priority Gas Auction (PGA) Modeling

To shed light on the strategies we observe in practice, we present a formal model for PGAs in this section. We then explain the notion of *advantage* that we use to understand strategies' effectiveness, and study several classes of observed strategies in the context of our model.

A. Model properties

We model a PGA as a sequential game among a set of n players $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{n-1}\}$ who bid against one another to obtain a payoff of $\$1$.[5]

This game has the following properties, which model blockchain dynamics:

a) **Continuous time:** Players act in continuous time, rather than discrete rounds (as in typical extensive-form games). This is because blockchain networks are asynchronous.

b) **Imperfect information:** Players eventually see one another's bids, but not immediately, a feature modeling blockchain network *latency*. Player $\mathcal{P}_i$ observes other players' bids after some fixed time Δ_i. Players may have differing latencies, small ones conferring a competitive advantage.

We measure latency as a relative figure to the latency of the miners, which are typically the best connected nodes in the network. Thus $\Delta_i = 0$ indicates that $\mathcal{P}_i$ has a superior network position and observes other player's bids with no additional delay over that of the miners.

c) **All pay:** In PGAs, losing players pay gas costs for their failed transactions. Our model captures this cost by having a losing player $\mathcal{P}$ pay $\ell(\$\mathsf{b}_{\mathsf{last}})$, where $\$\mathsf{b}_{\mathsf{last}}$ is the last bid made by $\mathcal{P}$ and $\ell()$ is a *loss function*.[6] This type of all-pay auction in which players can submit multiple bids is known as a *dollar* auction [33], in which not only the winner,

[5]In this "dollar auction," $\$1$ represents a normalization of any payoff amount w.l.o.g., and includes gas costs for the winner.

[6]As we explain above, bots use various tricks to reduce ℓ.

but losers pay. In our study, we typically observe auctions that are *partial all-pay*, i.e., $\ell(\$\mathsf{b}_{\mathsf{last}}) < \$\mathsf{b}_{\mathsf{last}}$.

d) **Probabilistic auction duration:** The auction terminates at a randomly determined time, namely when the next block is mined. For proof-of-work blockchains such as Ethereum, we model a block interval as a random variable D, which is exponentially distributed.

e) **Rate-limited bidding:** A player cannot raise her own bids continuously, but must wait a short interval δ to do so. This reflects throttling that blockchain peer-to-peer networks perform to prevent flooding attacks. As miners are always free to include the highest paying transaction that they observed, players can only raise bids, not lower them (a natural auction feature that holds in Ethereum PGAs).

f) **Minimum starting bid:** While there is no required minimum gas price in Ethereum, in practice PGA participants will want to start their bid at a price that gives them a good chance of getting included in the next block. Put another way, if a one bids too low, then even if nobody else bids against them, the miner may not include the transaction in their block. Thus, although not strictly enforced, we model our auctions as having a minimum starting bid, s.

g) **Minimum bid increments:** Players do not have freedom to raise bids by arbitrary increments. Instead, there is a minimum bid raise, ι, measured as a function of the player's previous bid. This matches Ethereum dynamics in which one can replace their transaction with another one, but it will only be relayed by the peer-to-peer network if the gas price has increased by a minimum threshold. Parity, the most common Ethereum node software enforces a default minimum increment of 12.5%.

Importantly, the bid increment restriction for $\mathcal{P}_i$ is entirely a function of the $\mathcal{P}_i$'s previous bid, but there is no enforced relationship between the bids of competing players. That is, players can outbid one another by arbitrarily small quantities so long as they are sufficiently raising their own previous bid.

In practice, network latencies and transaction-interval times are stochastic, but we assume constants Δ_i and δ to simplify our analyses. We believe that this simplification does not significantly affect strategy outcomes.

Continuous-time, imperfect-information games have seen only limited study. Most similar to our work is `FlipIt`, a continuous-time game with imperfect information that models advanced persistent threats (APTs) [36]. We are unaware of any prior work on such games with random play times or in general with the structure of PGAs.

B. Formal PGA model

For simplicity, we focus on games where $n = 2$, i.e., that involve a pair of players $(\mathcal{P}_0, \mathcal{P}_1)$. Our formalism, though, can be generalized to any n. The restriction to two players is supported by our empirical observations, which shows that most PGAs were played out between two players.

We let $b = (t, \$\mathsf{b}; i)$ denote a bid by player $\mathcal{P}_i$. Here t is the time at which the bid is placed, $\$\mathsf{b}$ is the bid price, and $i \in \{0, 1\}$ is the identity of the bidder.

We denote by $\mathbf{b}^*$ the sequence of all bids published to the network by all players at the current time t^*, and $\mathbf{b}$ the full sequence of all bids made by all players up to the point at which the auction ends. We let $\mathbf{b}[t]$ denote $\mathbf{b}$ at a particular time t. We let $\mathbf{b}_i$ and its variants (e.g., $\mathbf{b}_i[t]$) denote a bid sequence of player $\mathcal{P}_i$.

A *strategy* S_i for player $\mathcal{P}_i$ is a procedure for participating in a PGA, and may be probabilistic ("mixed," in game-theoretic terminology). S_i takes the following form, where input t^* is the current time and σ_i is the current local state of $\mathcal{P}_i$:

$$(a, t_{w,}, \sigma') \leftarrow\!\!\$\ S_i(\mathbf{b}^*, t^*_{,}\sigma).$$

The output a is an action by $\mathcal{P}_i$. Either $a = b$ for some bid b, or else $a = \perp$, indicating the player is not placing a bid. The output σ'_i represents an internal state update for the player.

Finally, $t_{w,\geq} t^*$ is a *wake time*. This is the time when $\mathcal{P}_i$ schedules its next execution *assuming no bids appear in the meantime*. The use of wake times means that w.l.o.g., we can assume that a player always emits a bid $b = (t^*, \$\mathsf{b}; i)$, i.e., for immediate publication. We also assume that when executed at an emitted wake time, a player always emits a bid; a player that chooses not to schedule a bid can set $t_{w,=}\infty$.

a) Game execution: A game between $\mathcal{P}_0$ and $\mathcal{P}_1$ involves an execution $\mathsf{Exec}(S_0, S_1, [D, \ell()])$ of their respective strategies. We specify the procedure for Exec in Figure 19 in Appendix C.

b) Payoffs: Players (bots) compete for financial rewards. We measure these rewards using the standard game-theoretic notion of a *payoff*. An execution $\mathsf{Exec}(S_0, \Delta_0, S_1, [D, \ell()])$ outputs a pair $(\$\mathsf{r}_0, \$\mathsf{r}_1)$. These are the respective payoffs (profits or losses) of $\mathcal{P}_0$ and $\mathcal{P}_1$.

We define the payoff of S_0 *against* S_1 as follows:

$$\mathsf{PO}^{\mathsf{Exec}}_{S_0,S_1}[(D, \ell())] = \mathbb{E}[\$\mathsf{r}_0 \mid (\$\mathsf{r}_0, \$\mathsf{r}_1) \leftarrow\!\!\$\ \mathsf{Exec}(S_0, S_1, [D, \ell()])]. \quad (1)$$

.

We refer to a strategy S_0 as *null-profitable* if $\mathsf{PO}^{\mathsf{Exec}}_{S_0,S_\varnothing}[(D, \ell())] > 0$ when $S_\varnothing$ is the null strategy (i.e., a strategy that never bids).

C. Why repeated bidding?

To understand the significance of our model, it is helpful to see why players' optimal strategies involve placing multiple bids over time. To show why, it is helpful by contrast to observe two settings in which players' optimal strategies involve placing a single bid.

a) Sealed-bid auction: Suppose that instead of the complex game just outlined, a PGA were a standard sealed-bid auction [31] (either first or second price, with $\ell() = 0$). Players bid once and ties are broken randomly.

Players would then be incentivized to cast a single bid as close as possible to 1, throwing away almost the entirety of their potential profit. This behavior is intuitive.

If bids are discrete, with tick sizes (smallest increments) of ϵ, then there is a Nash equilibrium for $S_0 = S_1$, the strategy of bidding $1-\epsilon$. In this case, $\mathsf{PO}^{\mathsf{Exec}}_{S_0,S_1}[(D,0)] = \epsilon/2$. This is the only Nash equilibrium for deterministic strategies, since a deviation that bids an extra ϵ would otherwise be profitable. For randomized strategies, non-cooperative players will converge towards this equilibrium after playing multiple games in which they study one another's behavior.

b) Fixed block duration: Returning to our model, when $\Delta_i, \Delta_j > 0$, placing multiple bids is a useful strategy *only when D is a random variable*. The reason is this: Given a block interval of fixed duration d, players can place the equivalent of sealed bids in the interval $[d-\delta, d]$ after the mining of the last block. Thus, in this setting the *only* reason for repeated bidding is on-chain signaling of intent, which is more expensive than out-of-band communication.

Notably, sealed-bid auctions are likely to be the norm on proof-of-stake blockchains, due to two reasons. First, in many proof-of-stake protocols the identity of near future miners is known in advance (perhaps anonymously, i.e., only the individual miner knows her timeslot). This implies that miners can accept bids over secure (encrypted and authenticated) out-of-band channels. Second, the block duration is quite predictable in proof-of-stake chains – miners forfeit their timeslot unless they broadcast their block before a limit (measured according to the local clocks of other miners) is reached. There is some measure of unpreditablility: the limit should be generous in order to accomodate propagation delays in the network, and miners may wish to collect many transactions that pay lucrative fees before broadcasting their block. Still, the block duration should behave according to statistical patterns, which is a far cry from the stochastic process of PoW blockchains. Our following analysis may thus apply to proof-of-stake systems with a high enough measure of unpredictability, though sealed-bid auctions are significantly more likely in such systems.

D. What's a good strategy?

Subject to their individual latencies, all players have identical strategy spaces, and thus we can directly compare strategies while only considering the latencies of the players that executed them. We define [Tyler: uhh this seems a bit abrupt, is some sentence missing here?]

$$\mathsf{Adv}^{\mathsf{exec}}_{(S_0,\Delta_0),(S_1,\Delta_1)}(D,\ell()) = \mathsf{PO}^{\mathsf{Exec}}_{S_0,S_1}[(D,\ell())] - \mathsf{PO}^{\mathsf{Exec}}_{S_1,S_0}[(D,\ell())].$$

VI. Priority-Gas-Auction Strategies

We now report on classes of strategies that we have observed empirically and examine them within the context of our model. Given the large strategy space, we restrict our focus to the most prevalent PGA strategies that we have observed.

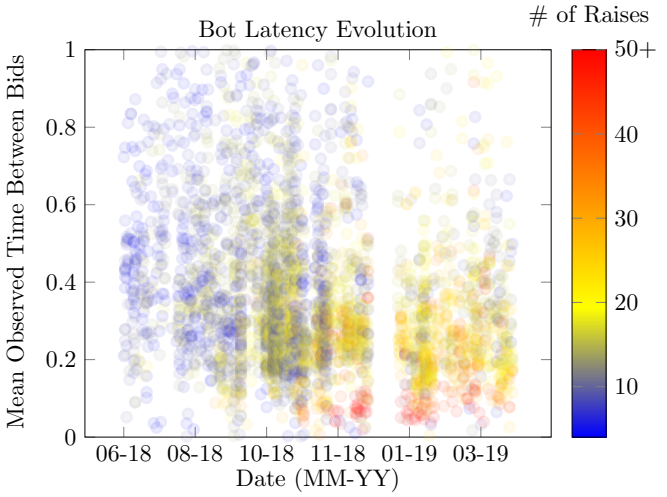


Fig. 9. Evolution of the PGA bots' latency strategies over time. Each point is the mean latency of a bot in some observed auction on the plotted date. The coloring of dots shows the number of total raises each bot placed in that respective auction. Early in the observed market, the majority of auctions contained 0-10 raises per bot. Over time, these auctions increasingly contained more raises and a lower mean latency. We exclude all observed mean latencies that do not fall between 0 and 1 seconds; there are such latencies, which are the result of the limitations of our monitoring architecture mentioned in IV-A1. The number of opportunities per bot, however, illustrates a trend that is far less vulnerable to instrumentation limitations, as it does not depend on the order or precise timing of observed transactions.

We thus consider three classes of strategies. The first two strategies are *blind raising* and *counterbidding*, which are often used in competition with one another. We also consider a *cooperative* strategy in which players take turns bidding. While we don't claim that we are seeing perfect cooperation (since if we were there would be no need for on-chain bidding at all), we show the cooperative strategy approximates and predicts a large class of behaviors that we have observed on-chain. Finally, we show that under conditions consistent with the PGAs we are observing on Ethereum, there exist Nash equilibria for cooperative strategies.

A. Latency Wars

For all strategies that we consider, our model predicts that players with a lower latency will have an advantage. Consistent with this feature of our model, we have observed empirically that players are consistently and substantially reducing their respective latencies over time (Figure 9).

B. Blind raising

Blind raising is a simple *non-adaptive* strategy. Player $player_i$ raises her own bids under a predetermined schedule, and the strategy is invariant to $\mathbf{b}^*_{1-i}$, the history of the other player's bids. In PGAs that we have observed, this schedule often involves repeated increases by fixed fractional increments (e.g., 12.5%, 21% or 70%). Basic blind raising is thus also *deterministic* ("pure"). A basic version may be modeled as follows:

- *Blind raising (S_0):* $\mathcal{P}_0$ emits a bid with amount $\$\mathsf{b}_0$ at time $\tau = 0$ and $\$\mathsf{b}_0 \times (1+f)^k$ at time $k\delta$.

Reactive counterbidding (S_1) :

- At time $\tau = 0$, $\mathcal{P}_1$ bids s.
- Denote the most recent bid by $\mathcal{P}_1$ as $\$\mathsf{b}_1$. $\mathcal{P}_1$ waits until it sees a bid $\$\mathsf{b}_0 > \$\mathsf{b}_1$ cast by $\mathcal{P}_0$. $\mathcal{P}_1$ immediately counterbids $\min(\max(\$\mathsf{b}_1 \times (1+\iota), \$\mathsf{b}_1 + \epsilon), 1 + \ell(\$\mathsf{b}_1))$.

Fig. 10. Reactive counterbidding PGA strategy. ϵ is the minimum bid tick, ι is the minimum bid increment, and s is the minimum starting bid. By bidding $\max(\$\mathsf{b}_1 \times (1+\iota))$, $\mathcal{P}_1$ ensures that its bid meets the minimum bid requirement and is also higher than $\mathcal{P}_0$'s bid. However, it would never make sense for $\mathcal{P}_1$'s bid to exceed $1+\ell(\$\mathsf{b}_1))$ as at this point it is more profitable to not bid and instead pay $\ell(\$\mathsf{b}_1)$.

At first glance, this strategy, and non-adaptive strategies in general, seem like a bad idea: They fail to exploit information available to the player in $\mathbf{b}^*$.

Surprisingly, however, the imperfect information created by network latency means that a *non-adaptive strategy can achieve an advantage over natural adaptive strategies*. The intuition is this: By playing non-adaptively, a player can publish a bid *faster* than by waiting to see the opposing player's bid and reacting to it. We explain below in particular how blind raising can achieve an advantage over a natural, adaptive strategy called *reactive* counterbidding.

C. Counterbidding

Counterbidding is a strategy in which a player observes an opponent's bidding strategy and reacts by placing higher opposing bids. A simple example of this strategy is what we call *reactive counterbidding*, an adaptive strategy that a player $\mathcal{P}_1$ might execute against another player $\mathcal{P}_0$. The strategy is described formally in Figure 10.

As $\mathcal{P}_0$ has interval $\geq \delta$ between bids, reactive counterbidding achieves an interval $\geq \delta$ between the bids of $\mathcal{P}_1$.

Many consensus algorithms (e.g., Nakamoto consensus) exhibit D exponentially distributed with mean λ. Again under the simplifying assumption that $\ell(\$\mathsf{b}) = \$\mathsf{c}$, we can show the following two observations, which we prove in Appendix C.

Observation 1. *Let $\Delta_i = 0$ and $\$\mathsf{r} = \mathsf{Adv}^{\mathsf{exec}}_{S_0,S_\varnothing}(D, \$\mathsf{c})$ be the payoff of null-profitable blind raising strategy S_0 against $S_\varnothing$. Then for*

$$\frac{\$\mathsf{c}}{\$\mathsf{r} + \$\mathsf{c}} < e^{-\lambda\delta},$$

as $\epsilon \to 0$, reactive counterbidding strategy S_1 has positive payoff.

We now show, somewhat counterintuitively, that with the right parameters—specifically, when Δ_1 is high w.r.t. δ — blind raising has an advantage over reactive counterbidding.

Observation 2 (Latency Amplification)**.** *For $\Delta_1 > \delta$, there exists a null-profitable, blind raising strategy, S_0, such that for any pure reactive counterbidding strategy S_1, S_0 achieves $\mathsf{Adv}^{\mathsf{exec}}_{(S_0,\Delta_0),(S_1,1)}(D, \ell()) > 0$.*

D. Cooperation

We now turn our focus to cooperative strategies, which we believe to be a close approximation to a common behavioral pattern we observe in which players slowly alternate raising bids. We demonstrate a cooperative Nash Equilibrium for the PGA game that helps shed light on the most common bidding behaviors that we observed.

Notice that as bids are non-decreasing, with each successive bid that players submit, the maximum profitability of the opportunity decreases. Intuitively then, it is beneficial for all players to decrease the total number of bids in the game.

It is natural then to assume that, whenever possible, players will coordinate out-of-band to split the profits rather than decrease their profits by competitively bidding on-chain. However, by the very virtue of us seeing on-chain PGAs, we know that perfect cooperation does not exist.

For example, if we consider a repeated game in which the same set of players is bidding on multiple (say for simplicity, equal value) opportunities, they can coordinate off-chain so that each opportunity only has a single bid on-chain and the profit for bidders is maximized. Indeed, in our measurement, we found many arbitrage opportunities for which there was only a single player, and it is entirely possible that off-chain coordination is at play here, although by its very nature there is insufficient data on chain to confirm this.

While off-chain cooperation may make sense, particularly in a repeated game, it has some drawbacks including lack of anonymity. We restrict our analysis here to the more interesting case of single game instances where players are bidding on-chain and coordinate their strategies to maximize expected profits Chief among them is that off-chain coordination is cumbersome and requires participants to identify themselves and sacrifice anonymity. This allows us to gain insight into the on-chain competitive auctions that we have observed.

As we will see, the cooperative strategies that we study help explain the behaviors in many of the PGAs that we have observed. Moreover, we show that under suitable parameters, there is a Nash Equilibrium for both players to follow a cooperative strategy.

We stress that for our cooperative equilibrium to emerge, it's not necessary that players explicitly coordinate out-of-band, but this behavior can develop organically on-chain. Indeed, we have seen evidence that over time, players have moved closer toward a cooperative equilibrium, which is consistent with well known results in experimental game theory that participants in the wild will converge to an equilibrium over time [30, 10].

a) Grim trigger equilibrium: We now describe a particular cooperative strategy in which any defection is responded to by immediately bidding the maximum amount, thereby eliminating all profitability from the auction.

This strategy class is similar to the *grim trigger* strategy that appears in the game theory literature in the context of repeated games [32]. Notice that since PGAs involve successive opportunities to bid potentially in response to the bids of other players, even a single iteration of our auction has elements of a repeated game.

Cooperative (S_i) with parameters D, W:

- At time $\tau = V[2k+i]$, $\mathcal{P}_i$ emits a bid with amount $W[2k+i]$.
- If $\mathcal{P}_i$ observes a bid of value $\$b$ at time t such that $t < V[2k+i]$ but $\$b > W[2k+i-1]$, then $\mathcal{P}_i$ immediately bids $\$1 + \ell(\$b)$.

Fig. 11. Grim-trigger cooperative PGA strategy

Consider two players who coordinate with the following strategies: at set intervals, players alternate their bids, each time bidding the minimum increase. If any player deviates either by bidding too soon or by raising the bid higher than they are supposed to, the other player will respond by raising their bid such that the game is no longer profitable. This strategy is characterized by following parameters: V is the set of agreed upon times at which bids will be scheduled, and W is the set of agreed upon bid values, such that that $W[i]$ will be bid by the appropriate player at time $V[i]$.

The grim-trigger cooperative strategy is formally described in Figure 11.

Notice that in the second condition, we assume that $\mathcal{P}_i$ can determine the time t that the bid was emitted by $\mathcal{P}_{1-i}$. This is consistent with our model in which the latency of each player is known. Thus, even though $\mathcal{P}_i$ will not observe $\mathcal{P}_{1-i}$'s bid until time $t+\Delta_i$, it can compute t by subtracting Δ_i from the time that it first observed the bid.

As we will see, the knowledge gap due to latency is an important feature when determining if our model has an equilibrium. All defections will eventually be detected, but depending on the latencies, there may still be sufficient time to profit from deviating.

b) Minimum bid raises: In a cooperative PGA, the main function of bidding on-chain is to alternate bids between players, but all players are incentivized to keep the actual bid price as low as possible to maximize profit. The following observation follows:

Observation 3. *In a grim-trigger cooperative PGA, for all $i \leq i_{max}$, the optimal choice for bids is $W[0] = s$ and $W[i] = W[i-1] \times (1+\iota)$.*

Remarkably as shown in Figure 12, in our observed PGAs, players converge over time to a minimum bid raise of 12.5%, which is the minimum allowed raise in the Parity client. It is known that players in the wild will converge to Nash Equilibria [10, 30] and this on-chain behavior is consistent to a convergence toward elements of cooperative equilibrium.

We stress that we don't claim that perfect cooperation is occurring or even that our exact equilibrium is on play. We do claim, however, that players are converging to a more cooperative state, that is approximate by our model, in which they allow opportunities for other players in an effort to maximize their own expected profit over time.

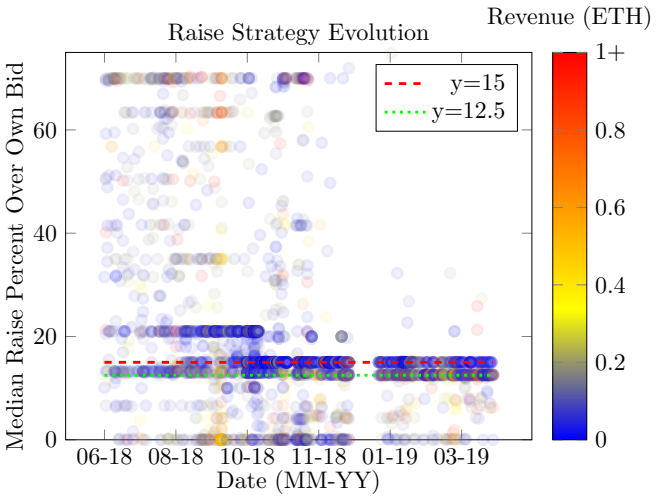


Fig. 12. Evolution of the PGA bots' median raise strategies over time for bots with more than 4 bids in pure revenue PGAs. Note the strategy convergence in the iterated game to almost exclusively the 12.5 (minimum raise) and 15% (deviation from minimum raise) levels. Opportunities below the $y = 12.5$ line are likely artifacts of bots emitting multiple transactions from globally distributed nodes; these opportunities correspondingly show a bias towards high revenue compared to those using the predicted strategies. We trim outliers and omit points with median raises over 75%; we observe 5 such high-raise outliers of 4,007 total observations.

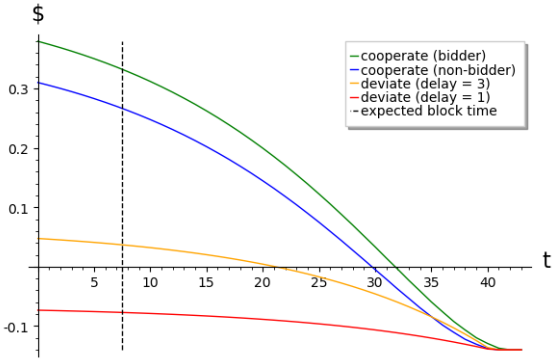


Fig. 13. Cooperative PGA with interval $t = 2$

c) Nash equilibrium: We now present our main result showing that there exist Nash equlibria for a wide range of parameters using the grim-trigger cooperative strategy.

Theorem 1. *For parameters consistent with Ethereum PGAs, there exist a grim-trigger Nash equilibria for a 2 player PGA where both players follow a D, W cooperative strategy.*

Figure 13 and Figure 14 show the expected payoff of the bidder and non-bidder in a cooperative PGA. The x-axis indicates the starting time of the interval, where the expected duration of the game is $1/\lambda = 15$ seconds. The y-axis indicates the expected payoff. Note that the players are alternating: if Player 1 is the bidder and Player 2 is the non-bidder when the first interval starts, then Player 1 is the non-bidder and Player 2 is the bidder when the second interval starts. These figures also display the profitability of deviation, i.e., the expected payoff of the non-bidder (according to latency parameters) in case

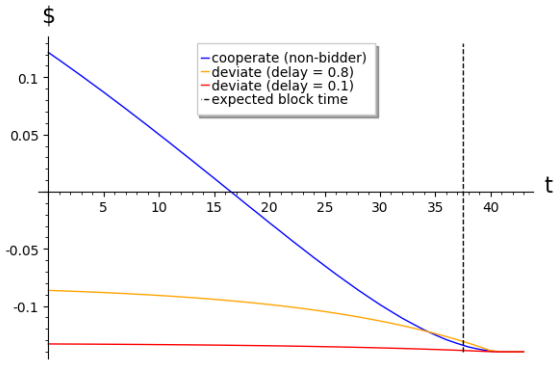


Fig. 14. Cooperative PGA with interval $t = 0.4$

she deviates when the interval begins.

If the deviation payoff surpasses the non-bidder payoff at any point (as with the `delay=3` plot in Figure 13), then a Nash equalibrium cannot hold for those parameters – a backward induction argument implies that if it is profitable for the non-bidder (say, Player 1) of interval i to deviate, then it must also be profitable for the non-bidder (Player 2) of interval $i - 1$ to deviate, and so on.

Figure 14 corresponds to the Ethereum parameters per our empirical measurements: $t = 0.4$ seconds is the mean interval length and $\delta = 0.1$ seconds is the estimated latency in our measurements.

To prove our main claim, let us recall that each player's bid increments her previous bid by 12.5%. Given our measurements, this implies 17 rounds before the players give the entire auction's reward to the miners. We can thus verify that in each of the few dozens of intervals prior to the end of the game, the expected payoff of adhering to the cooperative strategy is larger than the expected payoff upon deviation. Indeed, the Ethereum parameters that are plotted in Figure 14 show that the expected non-bidder payoff is always greater than the expected payoff of deviation. Theorem 1 thus follows.

VII. Miner-Extractable Value and Blockchain Security

PGAs and DEX arbitrage may not seem immediately harmful or relevant to the security of an underlying blockchain. They might simply seem an efficient mechanism for conveying market information among network participants. Unfortunately, we now argue, DEXes in fact present a serious security risk to the blockchain systems on which they operate, i.e., *at the consensus layer*. In other words, an key result of our work is that *application-layer security poses a current and direct threat to consensus-layer security*.

In a stable blockchain, block rewards incentivize honest miner behavior. As we show empirically in this section, though, *order optimization* (OO) fees, or implicit fees a miner is able to reap by leveraging their control of a consensus epoch, can exceed the block reward and instead incentivize forking attacks. These fees represent what we call *miner-extractable value* (MEV), which can subsidize forking attacks of two different forms. The first is a previously shown *undercutting attack* [11] that forks a block with significant MEV. The second is a novel attack, called a *time-bandit attack*, that forks the blockchain retroactively based on past MEV.

Undercutting attacks were previously considered a risk primarily in the distant future, when block rewards in Bitcoin are expected diminish below transaction fees. By measuring the significance and magnitude of OO fees, our work shows that undercutting attacks are a *present threat*.

Time-bandit attacks are also a present and even larger threat. They can leverage not just OO fees, but *any* forms of miner-extractable value obtained by *rewinding* a blockchain. Time-bandit attacks' existence implies that DEXes and many other contracts are inherent threats to the stability of PoW blockchains, and the larger they grow, the bigger the threat.

A. OO fees: Measurement study

We now lower bound the severity of OO fees in DEXes.

Figure 15 shows the distribution of pure revenue OO fees as a fraction of total miner-extractable value in Ethereum. We conservatively estimate MEV in this context as the sum of explicit transaction fees and pure revenue OO fees, making our distribution an underestimate of the share of MEV that does not come from explicit fee payments in Ethereum. Note that PGAs themselves are a mechanism for miners to claim OO fees: if miners instead claimed OO fees directly, there would be no incentive to bid in or create a PGA.

Of all blocks since block 3,875,490 [Tyler: I think you need to state "as of this writing, at block height XXXXX" here] (the first block we observed with a pure revenue OO fee from decentralized exchange arbitrage), approximately 3.6% contain at least one pure revenue arbitrage transaction. Of these blocks, 17,897 blocks contain pure revenue OO profits that are $< 1\%$ of the total stealable fees, confirming our earlier conclusions that the most frequent pure revenue arbitrage opportunities are for small price differences in the course of normal trading across exchanges. Nonetheless, many blocks do contain substantial pure revenue fees. 36,860 observed blocks contain pure revenue fees that exceed 20% of total stealable fees, 12,002 blocks derive the majority (over 50%) of stealable fees from pure revenue, and 2,517 blocks derive more than 80% of stealable fees from pure revenue.

Occasionally, these fees can be quite high, and can provide substantial short-term miner incentives to orphan blocks or otherwise deviate from the mining protocol. Figure 16 shows the 20 blocks observed on Ethereum with the highest absolute pure revenue OO fees.

As an example, the highest block, block 7029147 (https://etherscan.io/block/7029147), contained an arbitrage trade generating a revenue of 101.6 ETH, dwarfing the block reward of 3 ETH and the insignificant explicit transaction fees of 0.022 ETH. In this particular block, one transaction generated all the pure revenue OO fees. This transaction, whose profit graph is available at https://bit.ly/2WHGD0z, saw an exchange of Bigbom token (BBO) for ETH on Bancor and Kyber; an

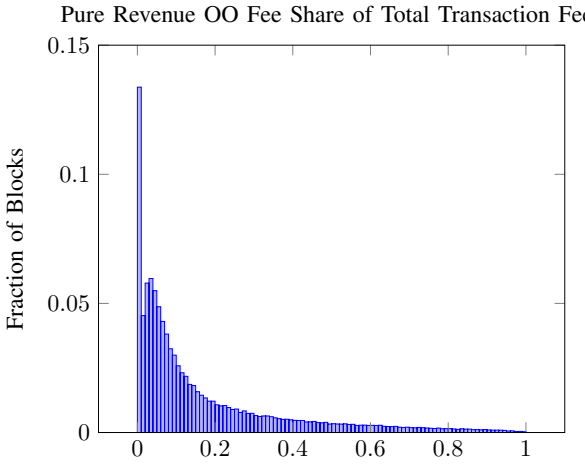


Fig. 15. Distribution of pure revenue OO as percentage of block transaction fees in blocks containing such transactions on the Ethereum network, blocks 3875490 to 7408826; 133,815 of 3,533,336 contain observed pure revenue (3.6%). Since block 7,000,000 (Jan 2 2019), this has increased to 6.3%.

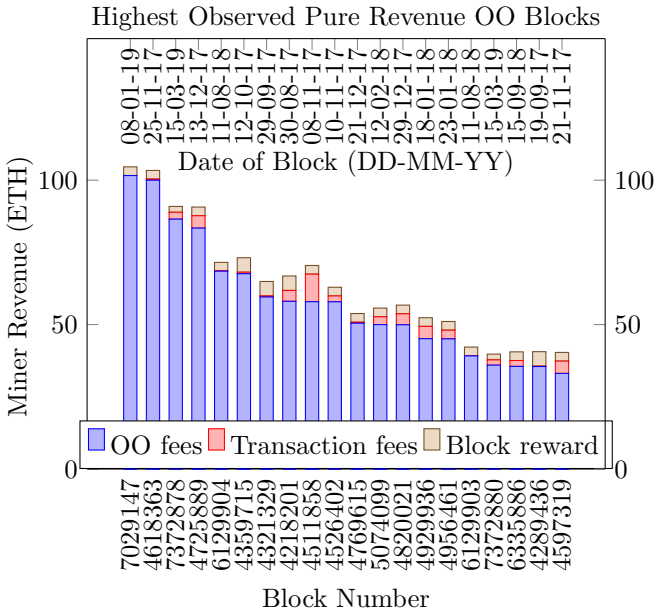


Fig. 16. Blocks with the highest pure revenue OO fees observed on Ethereum. As is shown here, OO fees in these blocks dominate both block rewards and transaction fees, often by more than an order of magnitude.

error in Bancor's pricing formula allowed a user to buy BBO at rates far under market. The bot's order size was 101.6 ETH, of which almost all was taken as pure profit.

As with all our analyses, we stress that our measurements are conservative: They represent lower bounds on arbitrage behavior. The limitations in our instrumentation harness discussed in Section IV-A1, lead to potential underestimation.

B. Undercutting attacks

Undercutting attacks [11] represent one vector of attack that can leverage OO fees.

It is well known that fixed per-block miner rewards are a key feature of secure and stable cryptocurrency protocols, as originally described in [11]. In that work, the authors analyze the incentives of blockchain miners in a regime where transaction fees exceed the inflationary subsidy paid to miners by a blockchain protocol. They observe that when the block reward is dominated by fees, rewards have high variance. As a result, a miner can fork a high-fee block, holding back some fees to attract other miners to build on the fork. In extreme cases, incentives to deviate from the protocol may lead to disruption in miner strategies for economically rational miners, reducing the security provided by block confirmations.

Prior to our work, transaction fees have been viewed as the only source of value for undercutting attacks, with [11] even calling a world in which fees dominate block rewards a "transaction-fee regime." That regime seemed a distant prospect. The title of [11], "On the Instability of Bitcoin Without the Block Reward," refers to Bitcoin block rewards going to zero, an event anticipated around the year 2140.

Our study shows that OO fees are a form of value that sometimes dominates explicit transaction fees *today*. The tail of the distribution in Figure 15 (transaction fee proportions > 0.5) and all of the example blocks in Figure 16 represent such opportunities. In other words, undercutting attacks represent a *present threat* in Ethereum, and one that will grow with the success of smart contracts that attract OO fees. Other potential sources of OO fees are mentioned in [12].

Most importantly, our view of arbitrage remains conservative, and we emphasize that pure revenue opportunities represent a small slice of total ordering fees. There are *many* possible sources of ordering fees payable to miners, including more sophisticated arbitrage. There are also many other forms of miner-extractable value. Miners can "steal" arbitrage opportunities from arbitragers by taking them themselves. Additionally, the survey in [14] describes several other sources of MEV, including the ability to buy into profitable ICOs early and the manipulation of games of chance.

Undercutting attacks can leverage OO fees or any other MEV from newly generated blocks. The second attack we describe can use MEV from *past blocks* to subsidize an attack.

C. Time-bandit attacks

We now describe *time-bandit attacks*, a new, second attack vector that can exploit miner-extractable value. They can exploit MEV from new blocks, but more powerfully, can also use MEV from *past blocks* via rewinding.

Time-bandit attacks are conceptually simple. Suppose a blockchain has a suffix (subchain) $[\mathsf{height}_0, \mathsf{height}_1]$, with current block height height_1, in which stealable value exceeds block rewards. An adversary can rewind to height_0 and use the resulting MEV to subsidize a profitable 51% attack that mines a fork up to or past height_1.

Of course, a time-bandit attack relies on real-time access to massive mining resources. As noted in [5], however, "rental attacks" are feasible using cloud resources, particularly for systems such as Ethereum that rely heavily on GPUs, which are standard cloud commodities. Sites such as http://crypto51.app/

estimate the costs. We illustrate with an example that relies on MEV from DEX rewinding.

Example 1. *Consider a price spike from 1USD to 3USD in a token that trades on an on-chain automated market maker (e.g., Bancor [1]). A miner performing a time-bandit attack can now rewrite history such that it is on the buy side of every trade, accruing a substantial balance in such a token at below market rate*—before the price spike.[7]

For example, if the attacker wishes to rewrite 24 hours of history, and 1M USD of volume occurred in exchanges with rewritable history in that token, then the attacker can obtain a MEV gross profit of 1M $\times$ (3USD - 1USD) = 2M USD.[8]

At the time of writing (March 2019), http://crypto51.app/ estimates a 24-hour 51% rental-attack cost on Ethereum of about 1.78M USD, implying a net profit of around 220K USD.

We stress that time bandit attacks are *not limited to MEV from DEXes*. A variety of smart contract systems allow anyone to participate and earn profit for doing so, often a desireable design goal for the style of permissionless and open interaction that lends itself naturally to blockchains. Any on-chain action in the past that could have potentially profited a miner today, including actions that unconditionally earn them ETH in the past, are thus potential sources for time-bandit attacks. Because smart contracts are Turing complete scripts and carry complex interactions, estimating the size of these opportunities is a challenging problem we defer to future work.

Time-bandit attacks in Ethereum: Recent transaction statistics suggest that Ethereum is vulnerable to time-bandit attacks. For example, decentralized exchange volumes show a peak of 1.5 billion USD of traded assets on Ethereum's decentralized exchanges in July 2018 [18]. While it is hard to gauge the total stealable value in this volume, the current estimated one-month cost of a 51% attack on Ethereum according to http://crypto51.app/ is approximately 56 million USD, more than *25 times lower than this DEX volume*.

We posit that the OO fees alone that we have described threaten the security of today's Ethereum network. As Figures 15 and 16 show, blocks with high OO fees and/or arbitrage opportunities can already enable such attacks.

More generally and alarmingly, time-bandit attacks can be subsidized by a malicious miner's ability to rewrite profitable trades retroactively, stealing profits from arbitrageurs and users while still claiming gas fees on failed transactions that attempt execution. The resulting MEV is potentially massive, suggesting a possibly serious threat *in Ethereum today*.

Of course, a full analysis of the threat would require an understanding of how time-bandit attackers might compete against one another to harvest MEV—by analogy with PGAs. This is a topic for future research.

[7]The attacker can additionally use new user trades at the market 3USD price in the automated market maker system to offload their tokens into ETH.

[8]More sophisticated reordering strategies stand to profit the miner even more. The miner can selectively include orders that manipulate the prices received by other traders *in the past*.

VIII. Open Questions and Future Work

Our results raise many important questions, some about the arbitrage community itself. For example, it would benefit arbitrageurs to collude with miners, but we observe no such collusion: Preliminary experiments show that bot transactions are equally distributed across mining pools. Are there incentives to avoid collusion, such as concern about the exogenous impact of miner malfeasance coming to light?

Other questions arise from PGA modeling. Are PGAs positive- or negative-sum games? In what ways could our proposed model be helpfully enriched?

More broadly, it is important to observe that DEXes are just the tip of the iceberg. At the time of writing, IDEX, the largest, is ranked #119 by volume by coinmarketcap.com. It has about 1M USD in 24h volume, compared with 970M USD for Binance, the leading centralized exchange.

Centralized exchange malfeasance might seem not to impact on-chain security, but the two are inextricably linked. By altering token trading dynamics on-chain, an adversary could manipulate and profit from centralized exchange dynamics. For example, a time-bandit attack could accumulate cheap tokens for offloading in a centralized exchange.

These observations raise several key follow-up questions:

- What forms do arbitrage take in centralized exchanges and at what volumes? And frontrunning?
- Barring direct data access, what techniques can be used to accurately measure various forms of trading activity in centralized exchanges?
- What financial incentives do centralized exchanges create for malfeasances in DEXes? How might they impact blockchain stability?
- What other insights might our data yield on DEX arbitrage, especially in quantifying opportunities that are not pure revenue? How much larger is the full arbitrage economy than executed? Can tight bounds be developed for the amount of MEV on Ethereum today?

IX. Conclusion

We have reported on a sizable economy of bots profiting from opportunities provided by transaction ordering in DEXes. We quantify the breadth of a specific subset of arbitrage, *pure revenue opportunities*, providing lower bounds on the profitability of ordering manipulation.

We have also formally modeled the behavior of bots competing against each other for miner-supplied transaction priority in *priority gas auctions*. Our empirical study validates several key predictions of our model, including the convergence of bots on a form of profitable cooperation involving minimal gas-price increases. We also show that in many concrete cases, bots' revenue from pure revenue arbitrage alone far exceeds the Ethereum block reward and transaction fees.

Finally, we argue that *miner extractable value*, particularly in the form of *order optimization fees*, implicit fees from modifying transaction order, threatens blockchain consensus stability. Such fees are large enough to subsidize serious

attacks on the network. They constitute an economic vulnerability that should be a current cause for concern in Ethereum.

References

[1] *Bancor Network*. www . bancor . network. Referenced March 2019.

[2] Iddo Bentov et al. *The Cost of Decentralization in 0x and Etherdelta*. http://hackingdistributed.com/2017/08/13/cost-of-decent/. Aug. 2017.

[3] Henry Berg, Todd A. Proebsting, et al. "Hanson's automated market maker". In: *Journal of Prediction Markets* 3.1 (2009), pp. 45–59.

[4] Dan Bernhardt and Bart Taub. "Front-running dynamics". In: *Journal of Economic Theory* 138.1 (2008), pp. 288–296.

[5] Joseph Bonneau. "Hostile blockchain takeovers (short paper)". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2018, pp. 92–100.

[6] Lorenz Breidenbach, Philip Daian, and Florian Tramer. *GasToken*. https://gastoken.io/. Mar. 2018.

[7] Eric Budish, Peter Cramton, and John Shim. "The high-frequency trading arms race: Frequent batch auctions as a market design response". In: *The Quarterly Journal of Economics* 130.4 (2015), pp. 1547–1621.

[8] V Buterin. "Schellingcoin: A minimal-trust universal data feed". In: *Ethereum Blog* (2014).

[9] Vitalik Buterin. *Hard Fork Completed*. https://blog.ethereum.org/2016/07/20/hard-fork-completed/. July 2016.

[10] Antonio Cabrales, Rosemarie Nagel, and Roc Armenter. "Equilibrium selection through incomplete information in coordination games: an experimental study". In: *Experimental Economics* 10.3 (2007), pp. 221–234.

[11] Miles Carlsten et al. "On the instability of bitcoin without the block reward". In: *ACM CCS*. 2016, pp. 154–167.

[12] Jeremy Clark et al. "On Decentralizing Prediction Markets and Order Books". In: *WEIS*. 2014.

[13] Phil Daian. *Analysis of the DAO exploit*. http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/. June 2016.

[14] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. "SoK: Transparent Dishonesty: front-running attacks on Blockchain". In: *arXiv preprint arXiv:1902.05164* (2019).

[15] Etherscan.com. https://etherscan.io/chart/blocktime. Referenced Apr. 2018.

[16] Gianni Fenu et al. "The ICO phenomenon and its relationships with ethereum smart contract environment". In: *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE. 2018, pp. 26–32.

[17] Neil Gandal et al. "Price manipulation in the Bitcoin ecosystem". In: *Journal of Monetary Economics* (2018).

[18] Georgi Georgiev. *Decentralized Exchange Trading Volume Hits All-Time Low*. https://bitcoinist.com/decentralized-exchange-trading-volume-low/. Jan. 2019.

[19] Ari Juels, Ahmed Kosba, and Elaine Shi. "The Ring of Gyges: Investigating the future of criminal smart contracts". In: *ACM CCS*. 2016, pp. 283–295.

[20] Olga Kharif. "CryptoKitties mania overwhelms Ethereum network's processing". In: *Bloomberg* (4 Dec. 2017).

[21] Sunny King and Scott Nadal. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. 2012.

[22] Ahmed Kosba et al. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts". In: *IEEE S&P*. 2016, pp. 839–858.

[23] S Demian Lerner. *Rootstock: Bitcoin powered smart contracts*. 2015.

[24] Michael Lewis. *Flash boys: a Wall Street revolt*. W.W. Norton & Company, 2014.

[25] Loi Luu et al. "Making smart contracts smarter". In: *ACM CCS*. ACM. 2016, pp. 254–269.

[26] Viktor Manahov. "Front-Running Scalping Strategies and Market Manipulation: Why Does High-Frequency Trading Need Stricter Regulation?" In: *Financial Review* 51.3 (2016), pp. 363–402.

[27] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. "A smart contract for boardroom voting with maximum voter privacy". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2017, pp. 357–375.

[28] Robert McMillan. "The inside story of Mt. Gox, Bitcoin's $460 million disaster". In: *Wired. March* 3 (2014).

[29] Jonathan Meng. "Understanding Gambling Behavior and Risk Attitudes Using Massive Online Casino Data". Dartmouth Univ., 2018.

[30] Rosemarie Nagel. "Unraveling in guessing games: An experimental study". In: *The American Economic Review* 85.5 (1995), pp. 1313–1326.

[31] Tim Roughgarden. *Lecture #2: Mechanism Design Basics*. http://theory.stanford.edu/~tim/f16/l/l5.pdf. Sept. 2013.

[32] Tim Roughgarden. *Lecture #5: Incentives in Peer-to-Peer Networks*. http://theory.stanford.edu/~tim/f16/l/l5.pdf. Oct. 2016.

[33] Martin Shubik. "The dollar auction game: A paradox in noncooperative behavior and escalation". In: *Journal of conflict Resolution* 15.1 (1971), pp. 109–111.

[34] Nick Szabo. "Formalizing and securing relationships on public networks". In: *First Monday* 2.9 (1997).

[35] *Uniswap Exchange Protocol*. www.uniswap.io. Referenced March 2019.

[36] Marten Van Dijk et al. "`FlipIt`: The game of 'stealthy takeover'". In: *Journal of Cryptology* 26.4 (2013), pp. 655–713.

[37] Will Warren and Amir Bandeali. "0x: An open protocol for decentralized exchange on the Ethereum blockchain". In: *URl: https://github.com/0xProject/whitepaper* (2017).

[38] Gavin Wood. *Ethereum: A secure decentralised generalised transaction ledger*. 2014. URL: http : / / yellowpaper.io/.

[39] Fan Zhang et al. *Paralysis Proofs: Secure Dynamic Access Structures for Cryptocurrencies and More*. IACR ePrint 2018/096. 2018.

[40] Fan Zhang et al. "Town Crier: An authenticated data feed for smart contracts". In: *ACM CCS*. ACM. 2016, pp. 270–282.

Appendix A
Smart-Contract-Enabled Complex Nondeterminism

[Phil: TODO these placeholders]

While the remainder of our work will focus on pure revenue opportunities, not all opportunities are pure revenue. One example is the transaction []. In this transaction, arbtirage bots took advantage of a security vulnerability in the automated market maker Bancor that allowed them to buy [] token at a far lower price than market price, buying []ETH worth of market-value token for []ETH, a net profit of approximately []ETH. The arbitrage bots may have sold the resulting tokens on either a centralized or decentralized exchange, making the other side of their trade invisible to the network. Here, the revenue comes not directly from executing a single atomic on-trade transaction, but instead uses the on-chain transaction as a component of a wider trading strategy across both centralized and decentralized exchanges.

While arbitrage bots' intent is often clear by inspection of their transactions, other times, intent may be opaque, nondeterministic, or may depend on unpredictable aspects of the Ethereum network state. The smart contract wrappers through which bots can execute transactions enable more than simple batching of transactions. Such contracts are actually atomically-executing programs written in Turing-complete scripting languages, and can therefore implement complex strategies.

For example, during, before, or after executing a set of trades, a wrapper contract can also query prices from and conditionally trade with automated market makers. Because the prices offered by these automated market makers cannot be known in advance, and because batches can revert, even based on changes in the Ethereum global state and the success of other batches, orders placed through smart contracts can express complex conditional preferences on trade execution. Complex trades can also perform arbitrary computation on the Ethereum network. Such complex order types do not have analogs in traditional HFT, but are similar to proposals such as [**conditionalorders**].

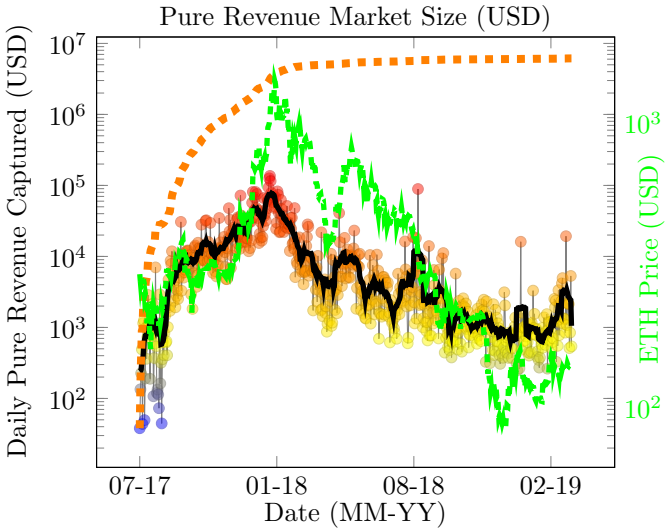


Fig. 17. todo

Appendix B
Additional Data Analyses

A. Pure Revenue USD Market

Figure 17 shows the ETH price and size of the pure revenue market in USD. When compared to the equivalent graph denominated in ETH (Figure 4), some effect of price reductions on the market size are evident. Opportunities in ETH are relatively consistent and well distributed, as we expect from mechanical rents extracted due to exchange design fundamentals. Such USD graphs appear more correlated with the price, showing decreased revenue during price slumps and contradicting our other data about the sophistication of the PGA bot market increasing over time.

Because of this and the relevance of ETH to direct protocol security analyses, we denominate the majority of our synthesized results in ETH.

B. Pure Revenue, Broken Down by Bots

Figure 18 shows the size of the full pure revenue opportunity market captured by the top 10 transaction senders; note that the graph is very similar to the graph of the estimated profits we calculate, having a relatively constant offset and almost identical shape for all market players.

This may reflect imperfections in our heuristics for calculating profit; we cannot include, for example, server costs, and it is difficult to tell which on-chain transactions other than a pure revenue transaction may have contributed to a bot's specific costs. We also do not include costs of failing to capture competitive pure revenue opportunities, which may be significant; it is difficult to disambiguate these failed transactions from unrelated transactions, as they may never execute or attempt to execute the intended trades. These matters are left to future work and analyses of the data we publish.

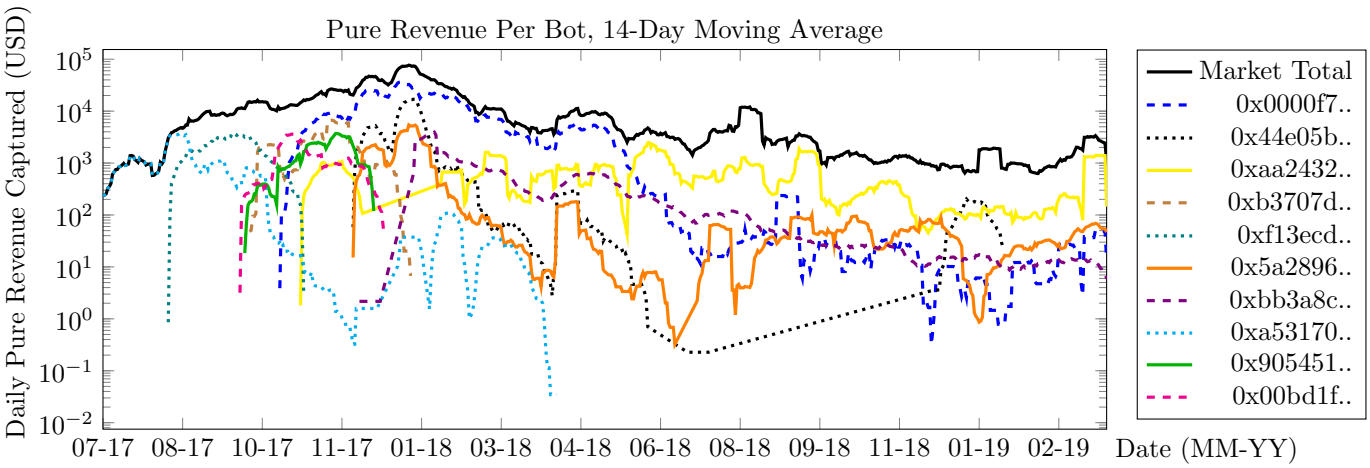


Fig. 18. Pure revenue bot breakdown, as described in Section **??**, showing revenue without subtracting transaction costs.

APPENDIX C
ADDITIONAL PGA MODELING DETAILS

Proof of Observation 1. S_1 counterbids after time δ. Since D has distribution $\mathsf{Exp}(\lambda)$, the probability that any bid by S_0 will have priority in a mined block is $\Pr[X < \delta]$, for X a random variable with distribution $\mathsf{Exp}(\lambda)$. This holds with probability $1 - e^{-\lambda\delta}$. Thus, the payoff of S_1 is positive if:

$$e^{-\lambda\delta}\$\mathsf{r} - (1 - e^{-\lambda\delta})\$\mathsf{c} > 0.$$

The observation follows. □

Proof of Observation 2. Since $\Delta_i > \delta$, S_0 can schedule its bids at intervals less than Δ_i. Thus, by the time $\mathcal{P}_1$ observes a bid $\$\mathsf{b}_i$, $\mathcal{P}_0$ has already cast a new bid $\$\mathsf{b}_{i+1}$. Since S_1 is pure, $\mathcal{P}_1$ can choose a value $\$\mathsf{b}_{i+1}$ that outbids $S_0's$ reactive bid to $\$\mathsf{b}_i$. Thus $\mathcal{P}_1$'s bids will always maintain the highest bid and win the PGA. □

Observation 4. *There exists a finite number of intervals i_{max} for a cooperative PGA after which bidding is no longer profitable, even if the block has not yet been mined.*

Proof. Let i_{max} be the greatest value for which the following inequality holds:

$$s \times (1 + \iota)^{i_{max}} \leq 1 + c,$$

where where s is the starting bid and n is the minimum bid increment as defined in Section V.

We can equivalently define a latest end time for the PGA $t_{end} = V[i_{max}]$. □

We define a function $F_t^{\mathcal{P}_i}(b)$, which denotes the expected payoff that a player $\mathcal{P}_i$ will receive by bidding $\$\mathsf{b}$ at time t given the strategies of all other players. In order for a Nash equilibrium to hold for a cooperative strategy, it must be the case that

$$\forall i, b', t' : F_{V[i]}^{\mathcal{P}_i}(W[i]) \geq F_{t'}^{\mathcal{P}_i}(b'),$$

where $V[i] < t' < V[i+1]$.

Observation 5. *Denote by $p_{time}(t)$ the probability that a PGA has ended by some time t. For proof-of-work blockchains, $p_{time}(t) = 1 - e^{-\lambda t}$.*

Proof. Since the block interval in a proof-of-work blockchain is exponentially distributed, we model the PGA end time as an exponential distribution with rate parameter λ. The probability that the auction has ended by some positive time t is given by the CDF of the exponential distribution:$1 - e^{-\lambda t}$. □

Observation 6. *Assume that miners continuously update their blocks with the most profitable set of transactions. For players $\mathcal{P}_0, \mathcal{P}_1$ adhering to the cooperative strategy with parameters D, W, the probability that $\mathcal{P}_b$ will win the PGA is given by*

$$\sum_{i=2k+b}^{i_{max}} (1 - p_{time}(V[i])) \times p_{time}(V[i+1]),$$

for $k \in [0, \frac{i_{end}}{2}]$.

Proof. Notice that probability that a player wins a PGA is exactly the probability that the PGA terminates during an interval controlled by that player. This follows since miners continuously update their blocks with new more profitable transactions and the player in control of the interval will have the highest bid during its interval.

We can define the probability that the game ends during a given interval as

$$p_{interval}(i) = (1 - p_{time}(V[i])) \times p_{time}(V[i+1]),$$

and the probability of $\mathcal{P}_b$ winning is the summation of $p_{interval}$ for all intervals that it controls. □

$\mathsf{Exec}(S_0, S_1, [D, \ell()])$

$\mathbf{b}^*, \mathbf{p} \leftarrow \emptyset; s_0, s_1 \leftarrow \emptyset; t^*, t_{w,0}, t_{w,1} \leftarrow 0$
$t_{\mathsf{end}} \leftarrow\!\!\$\ D$
do
  if $\mathsf{FirstBidTime}(\mathbf{p}) \leq \min(t_0^*, t_1^*)$ **then**
    $t^* \leftarrow \mathsf{FirstBidTime}(\mathbf{p})$
    $\mathbf{b}^* \leftarrow \mathbf{b}^* \cup \mathsf{PopFirstBid}(\mathbf{p})$
    for $i \in \{0, 1\}$ **do**
      $(a, \sigma_i', t_{w,i}) \leftarrow\!\!\$\ S_i(\mathbf{b}^*, \sigma_i, t^*)$
      $\sigma_i \leftarrow \sigma'; t_i^* \leftarrow t_{w,i}$
      if $a \neq \perp$ **then**
        $\mathbf{p} \leftarrow \mathbf{p} \cup a$
  if for $i \in \{0, 1\}, t_i^* < \mathsf{FirstBidTime}(\mathbf{p})$ and $t_i^* \leq t_{1-i}^*$ **then**
    $t^* \leftarrow t_i^*$
    $(a, \sigma_i', t_{w,i}) \leftarrow\!\!\$\ S_i(\mathbf{b}^*, \sigma_i, t^*)$
    $\sigma_i \leftarrow \sigma_i'; t_i^* \leftarrow t_{w,i}$
    $\mathbf{p} \leftarrow \mathbf{p} \cup a$
until $\mathsf{FirstBidTime}(\mathbf{p}) > t_{\mathsf{end}}$
$a_{\mathsf{win}} = (t_{\mathsf{win}}, \$\mathsf{b}_{\mathsf{win}}; i) \leftarrow \mathsf{MaxBid}(\mathbf{b}^*)$
$a_{\mathsf{lose}} = (t_{\mathsf{lose}}, \$\mathsf{b}_{\mathsf{lose}}; 1 - i) \leftarrow \mathsf{MaxBid}(\mathbf{b}_{1-i}^*)$
if $i = 0$
  output $(\$\mathsf{r}_0, \$\mathsf{r}_1) = (\$1 - \$\mathsf{b}_{\mathsf{win}}, \ell(\$\mathsf{b}_{\mathsf{lose}}))$
else
  output $(\$\mathsf{r}_0, \$\mathsf{r}_1) = (\ell(\$\mathsf{b}_{\mathsf{lose}}), \$1 - \$\mathsf{b}_{\mathsf{win}})$

Fig. 19. **Pseudocode for Exec, the execution of a PGA.** This procedure maintains a list of pending bids $\mathbf{p}$ awaiting transmission, i.e., subject to latency Δ [Steven]: I think we removed the outgoing latency and only consider incoming latency]. For clarity, we define several functions. $\mathsf{FirstBidTime}(\mathbf{p})$ is the time t associated with the earliest pending transaction. $\mathsf{PopFirstBid}(\mathbf{p})$ pops and outputs the earliest transaction; given two transactions with the same time, it picks one uniformly at random. $\mathsf{MaxBid}(b)$ outputs the maximum price bid in b; given multiple bids at the same price, it outputs the first in temporal order.

For any interval $i < i_{max}$, we can now define the expected payoff for each player during that interval, conditioned on the fact that the auction has already reached the interval $j \leq i$ and has not yet terminated. For each interval, we give two expected values: one for the "bidder" — i.e., the player whose turn it is to bid during that interval and one for the "non-bidder". The bidder's expected reward is given by:

$$E_{bidder}(i, j) = p_{interval}(i - j) \times (1 - W[i]).$$

The non-bidder's expected reward during a given interval is given by:

$$E_{non-bidder}(i, j) = p_{interval}(i - j) \times -c.$$

Now, we define the expected payoff for each player by continuing to follow the cooperative strategy at any point in the game. As in our definition of the strategy, we assume without loss of generality that $\mathcal{P}_0$ bids first and $\mathcal{P}_1$ bids second.

Observation 7. *During the jth interval, the expected payoff for $\mathcal{P}_b$ to continue to cooperate for the rest of the auction is given by:*

$$E_{cooperate}^{(\mathcal{P}_b)}(j) = \sum_{i=0,\infty} E_{bidder}(2i + j + b, j) + \sum_{i=0,\infty} E_{non-bidder}(2i + j + (1 - b), j) + (1 - p_{time}(V[i_{end}])) \times -c$$

Proof. This follow directly form the previous observation. Previously, we defined the expected value of a bidder and a non-bidder in any interval. If we take an infinite sum of these expected values, alternating between bidder and non-bidder corresponding to the intervals in which the respective players bid, then we will get the expected payoff for the game. □

Next, we need to show the cost of deviation. For the grim trigger cooperative strategy, each player knows that if they defect, the other player will raise the bid to make the game unprofitable – i.e., each player's payoff will be $-c$.

While intuitively this arrangement (assuming that the other player will make good on its threat) makes deviation unwise, deviation may still be profitable if the auction ends before the other player can respond. This can happen due to the other player's latency – i.e., if there is some time delay in which

the deviation goes undetected. Additionally, this can occur due to the rate limit – i.e., the deviation was detected, but the other player is not yet able to place a bid to make good on its threat. For the sake of this analysis, the latency and rate limit serve identical functions – namely, to delay the other players response. Thus for simplicity, we define

$$\mathsf{delay}^{(\mathcal{P}_b)} = \texttt{Max}[\Delta_{1-b}, \delta].$$

Observation 8. *If for all i, it holds that $\mathsf{delay}^{(\mathcal{P}_b)} < W[i+1] - W[i]$, then the expected payoff for $\mathcal{P}_{1-b}$ deviating during the ith interval is given by*

$$E^{(\mathcal{P}_b)}_{deviate}(\mathsf{delay}^{(\mathcal{P}_b)}, i) = p_{time}(\mathsf{delay}^{(\mathcal{P}_b)}) \times \texttt{Max}(-c, 1 - W[i+1]) + (1 - p_{time}(\mathsf{delay}^{(\mathcal{P}_b)}) \times -c$$

.

Proof. Since $\mathsf{delay}^{(\mathcal{P}_b)} < W[i+1] - W[i]$, then $\mathcal{P}_b$ will notice and be able to react to $\mathcal{P}_{1-b}$'s deviation during the current interval, at which point, $\mathcal{P}_b$ will react making the PGA unprofitable for both players.

Thus, $\mathcal{P}_{1-b}$'s expected profit is exactly the expected reward that it makes during the time before the deviation is noticed, which is given by the above equation. □

Lemma 1. *The cooperative strategy with parameters D, W will yield a Nash Equilibrium if $\forall j$,*

$$E^{(\mathcal{P}_b)}_{cooperate}(j) \geq E^{(\mathcal{P}_b)}_{deviate}(\mathsf{delay}^{(\mathcal{P}_b),j}).$$

Proof of Observation 1. This follows directly from the previous observations. □

Appendix D
OO Fees and Other System Designs

[Phil: TODO fix these references]

We now discuss high-level consequences of OO fees in the context of different types of blockchain systems being built in academia and industry. The complete enumeration of their impact is left to future work.

- **Proof-of-stake systems** In any proof of stake system where forks are allowed (e.g. [**snowwhite**] [21] [**cardano**]), similar incentives as for Proof of Work exist for miner to orphan or rewrite history when profitable OO fees exist. Regardless of whether sealed-bid auctions (cf. Section V-C) are prominent, the next miner may overtake the previous block and replay its transactions in her block in an optimized order. Some stake-based protocols, on the other hand, attempt to provide a notion of *finality*, and feature clients which will not revert history regardless of evidence presented. For systems in which the finality notion refers to a checkpoint that is done to cement multiple blocks that have already been created (e.g. [**ffg**]), all of the concerns remain the same, because rapid bidding wars affect the appeal of short-term (in particular, single block) forks. In blockchains that seek to finalize every block (e.g. [**algorand**, **tendermint**]), the potential of high profit OO fees implies that the honest (super-)majority assumption of such systems may not coincide with rational behavior.
- **Permissioned blockchain systems** Permissioned blockchains are currently being explored by many large financial institutions, often in use-cases like exchanges [**dexexchange**]. The importance of transaction order in these decentralized exchanges poses a variety of interesting questions for such systems. For example, it is classically claimed that blockchains add auditability to existing workflows, but it is impossible to audit or objectively divine the real order in which a block producer received transactions on an asynchronous network. Furthermore, because OO fees exist even on permissioned chains where on-chain assets are being exchanged, block producers in such chains must be chosen carefully and trusted with correct operation and ordering.
- **Sharded blockchain systems** Several sharded blockchain systems have been explored in both academia (e.g. [**omniledger**]) and industry (e.g. [**ethsharding**]). One important consequence of sharding blockchains is the reduced security of each shard over the security of the whole system. Generally, the effects of this reduced security are mitigated through random sampling from a large pool of potential validators, so an adversary would require substantial control of the pool to have a high chance of adversarially controlling a shard. Unfortunately, as OO fees show, the security needs of shards may not be homogeneuous; a shard that operates a large decentralized exchange must pay miners higher rewards to ensure stability than a shard without such an exchange by the analysis in [**princetonpaper**]. Because few sharded based systems are running in practice, substantial future work remains to fully enumerate potential attacks and their mitigations in sharded systems.
- **Other exchange designs** In other blockchain-based exchange designs, it may still be possible for miners of the underlying system to manipulate prices, either in real time or retroactively. This sets up similar incentives as those explored in this work. One example is for channel based networks with public watchtowers such as [**pisa**]; miners can simply participate in exchanges, collecting old states through public watchtowers, and can selectively publish profitable states in a history rewriting attack that works as above. Thus, it is not necessarily the case that the abstraction achieved by Layer 2 exchange systems is sufficient to prevent ordering attacks.