A^{2}MM:
Mitigating Frontrunning, Transaction Reordering and Consensus Instability in Decentralized Exchanges

Abstract—**The asset trading volume on blockchain-based exchanges (DEX) increased substantially since the advent of Automated Market Makers (AMM). Yet, AMMs and their forks compete on the same blockchain, incurring unnecessary network and block-space overhead, by attracting sandwich attackers and arbitrage competitions. Moreover, conceptually speaking, a blockchain is one database, and we find little reason to partition this database into multiple competing exchanges, which then necessarily require price synchronization through arbitrage.**

This paper shows that DEX arbitrage and trade routing among similar AMMs can be performed efficiently and atomically on-chain within smart contracts. These insights lead us to create a new AMM design, an Automated Arbitrage Market Maker, short A^{2}MM DEX. A^{2}MM aims to unite multiple AMMs to reduce overheads, costs and increase blockchain security. With respect to Miner Extractable Value (MEV), A^{2}MM serves as a decentralized design for users to atomically collect MEV, mitigating the dangers of centralized MEV relay services.

We show that A^{2}MM offers essential security benefits. First, A^{2}MM strengthens the blockchain consensus security by mitigating the competitive exploitation of MEV, therefore reducing the risks of consensus forks. A^{2}MM reduces the network layer overhead of competitive transactions, improves network propagation, leading to less stale blocks and better blockchain security. Through trade routing, A^{2}MM reduces the predatory risks of sandwich attacks by taking advantage of the minimum profitable victim input. A^{2}MM also offers financial benefits to traders. Failed swap transactions from competitive trading occupy valuable block space, implying an upward pressure on transaction fees. Our evaluations shows that A^{2}MM frees up 32.8% **block-space of AMM-related transactions. In expectation, A^{2}MM's revenue allows to reduce swap fees by** 90%**.**

We hope that our work engenders further innovation in the space of efficient and censorship-resilient exchanges, which by design democratizes MEV and *let the people trade*.

I. Introduction

Permissionless blockchains have portrayed their full strength when mediating financial assets among parties within censorship-resilient on-chain exchanges. One of the most popular exchange models is the Automated Market Maker [35], where a smart contract autonomously adjusts the price for supply and demand upon incoming trading requests.

In a perfect world, different financial exchanges would all offer the same price for the same asset at the same time — i.e., the exchanges should be perfectly synchronized. In reality, however, competing exchanges must necessarily synchronize their asset prices. High-frequency arbitrage bots are known to conduct transaction fee bidding contests, both on the blockchain's P2P network and on the consensus layer [22], [52], [53]. Transaction fee bidding obstructs the available bandwidth on the blockchain's P2P network [22], therefore hinders information propagation and hence negatively influences blockchain security [31]. MEV was also shown to incentivize miners to fork the chain. For example, a small rational miner with only 5% hashrate, will fork the Ethereum blockchain given an MEV opportunity yielding $4\times$ the block reward [52].

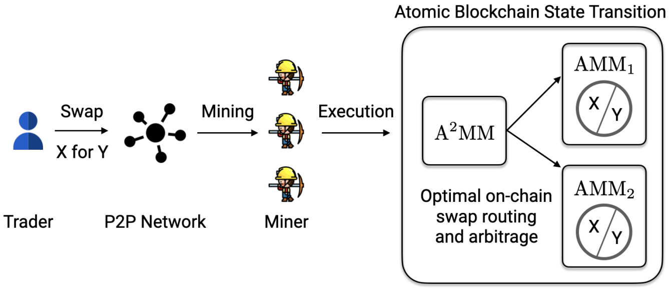
Atomic Blockchain State Transition
Swap
X for Y
Trader
P2P Network
Mining
Miner
Execution
A^{2}MM
Optimal on-chain swap routing and arbitrage
AMM$_1$
AMM$_2$
X
Y
X
Y

Fig. 1: A^{2}MM design, which peers with two AMMs using their liquidity pools. When A^{2}MM receives a swap transaction for a market with the assets X and Y, A^{2}MM atomically performs optimal routing and arbitrage among the considered AMM, minimizing subsequent arbitrage transactions.

This paper proposes a new type of AMM design, so-called Automated Arbitrage Market Maker, or A^{2}MM, which by design performs optimal trade routing and best-effort two-point arbitrage (cf. Figure 1) among peered AMMs. A^{2}MM offers various security benefits for the underlying blockchain. First, A^{2}MM atomically extracts two-point arbitrage MEV from the peered AMMs, which would otherwise deteriorate the blockchain's security [52]. Second, through swap routing, A^{2}MM reduces the risks of sandwich attacks due to the minimum profitable victim input (MVI) [53]. Third, A^{2}MM deters competitive network layer bidding, freeing the available blockchain network layer bandwidth, reducing the stale block rate and ultimately strengthening blockchain security [30].

A^{2}MM also offers financial benefits to its users. Routing grants traders better asset prices, and arbitrage can yield positive income. Contrary to centralized MEV relayer, which auctions off MEV extraction[1], *A^{2}MM is a decentralized and trustless design allowing users to atomically benefit from MEV, while mitigating its negative consequences*. One drawback of A^{2}MM is that the added smart contract logic necessarily increases the transaction fees for swaps. Our evaluation of A^{2}MM with two

[1] e.g., https://github.com/flashbots/mev-relay-js

AMMs (cf. Figure 1), however, shows that A^{2}MM's routing and arbitrage in expectation allow to reduce transaction fees by 90% compared to a standard AMM swap.

One Blockchain — One AMM

The AMM design space is without a doubt considerable and multi-dimensional [51]. Related works have for example explored the various implications of differing AMM pricing formulas [3], [1], [25]. Orthogonal to the pricing formula design space, we would like to propose an intuition of how multiple AMMs on the same blockchain can be positively united.

Conceptually speaking, a blockchain is a distributed database, where each blockchain node aims to maintain the same view as the remaining network. If we compare a blockchain to a centralized exchange, which must also maintain its proprietary non-distributed database, then there is little reason to split such a database into multiple competing partitions, which necessarily require synchronization through price arbitrage. We therefore observe the following: *(i)* multiple DEXes dilute the financial liquidity in each DEX and result in less attractive asset pricing. *(ii)* multiple DEXes must synchronize through arbitrage, which causes overhead on the blockchain database and the network layer. *From a security and financial perspective, it therefore appears to be strictly disadvantageous to deploy multiple DEXes on the same blockchain.*[2] In this work, we take the stance that a blockchain should ideally only operate one AMM smart contract, to increase the financial efficiency, reduce network layer and block-space overhead, and consequently increase blockchain throughput as well as security.

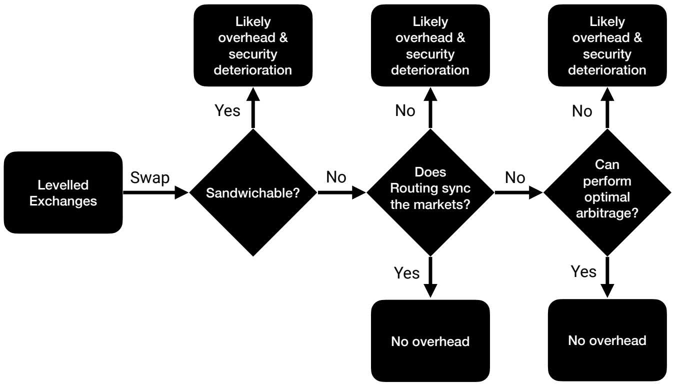
Levelled Exchanges
Swap
Sandwichable?
Yes
Likely overhead & security deterioration
No
Does Routing sync the markets?
No
Likely overhead & security deterioration
Yes
No overhead
No
Can perform optimal arbitrage?
No
Likely overhead & security deterioration
Yes
No overhead

Fig. 2: Decision tree whether a swap would introduce overhead in terms of the P2P network layer or block space.

We motivate our stance through an AMM swap decision tree in Figure 2. The decision tree departs from a possible AMM state where multiple exchanges on an identical blockchain have the same price for the same asset markets, i.e., the exchange prices are levelled. Then, a user performs a swap on one of the i AMM exchanges, which would necessarily depart the AMMs state from their price equilibrium. Given optimal sandwich attack parameters [53], we can determine whether the swap can be attacked through a sandwich attack. If the swap is not sandwichable, we find whether the AMMs can reach price equilibrium through optimal asset routing. If routing alone cannot level the prices on the different AMM markets, we can resort to arbitrage. If arbitrage can be performed atomically within the swap, we do not anticipate further overhead from the incoming swap, hence securing the blockchain from possible MEV extraction. In all other cases, there exist the likely introduction of overhead deteriorating the blockchain security, reducing its throughput and increasing transaction fees.

We summarize our main contributions in the following.

A^{2}MM Design: We provide a new AMM design, A^{2}MM, which atomically performs optimal swap routing as well as efficient arbitrage among existing AMM liquidity pools, if deemed profitable by the A^{2}MM smart contract. Our design does not change the usability of existing AMMs, yet allows any user to profit from routing and arbitrage.

A^{2}MM Strengthens Blockchain Security: MEV is a design problem threatening blockchain security [46], [52]. A^{2}MM mitigates two MEV sources, namely two-point arbitrages and sandwich attacks. We show that adopting an A^{2}MM design reduces both block-space and network layer overhead caused by MEV bots, therefore, strengthening the blockchain consensus by reducing the stale block rate. We find that 88.80% of the back-running arbitrage transactions are accompanied by what we call *back-run flooding* (BRF), an observed denial of service practice on the blockchain P2P layer.

Evaluation: We implement and evaluate A^{2}MM as shown in Figure 1, while synchronizing with two AMMs (i.e., Uni- and Sushiswap). By replaying past blockchain data, through routing and arbitrage revenue, on average, we find that A^{2}MM reduces the consumed transaction fees of a standard AMM swap by 90%. Moreover, in expectation, A^{2}MM reduces the consumed block-space by 32.8%.

The remainder of the paper is organized as follows. Section II provides a background, while Section III introduces our system and threat model and outlines the A^{2}MM design. Section IV presents our evaluation and empirical results. We highlight A^{2}MM's security implications in Section V and shed light on the cost when A^{2}MM peers with more than two AMMs in Section VI. We summarize related works in Section VII, provide a discussion in Section VIII and conclude the paper in Section IX.

II. Decentralized Finance (DeFi)

Since the inception of permissionless blockchains with Bitcoin in 2008 [45], it became apparent that their most well-suited use case is the transfer or trade of financial assets without trusted intermediaries [50]. A blockchain is considered permissionless when entities can join and leave the network at any point in time. Users authorize transactions through a public key signature and a subsequent broadcast on the blockchain P2P network. The formatted public key corresponds to an

[2] There may be social, competitive, and egocentric reasons for the deployment of competing DEXes (e.g., to sell a token), which we, however, do not further investigate in this work.

address that a user can receive assets at. Miners accumulate transactions and solve a proof of work (PoW) puzzle to append a block to the blockchain (various alternatives to PoW emerged, such as PoS [48], [12]). Miners are financially rewarded for performing work for the network through block rewards and transaction fees. A third miner reward source, which is gaining traction [46], is the extraction of Miner Extractable Value. While Bitcoin supports basic smart contracts through a stack-based programming language, the support for loops and higher-level languages (such as Solidity) have gained widespread adoption. For a more thorough blockchain background, we refer the reader to several helpful SoKs [18], [10], [11].

Smart contracts provide the building blocks for an ecosystem of decentralized finance [49], where users can interact with lending pools, AMM exchanges, stablecoins, derivatives, asset management platforms etc. At the time of writing, DeFi has grown to an accumulative locked value of over $60B$ USD. For a more thorough background on DeFi, we refer the interested reader to an SoK [49]. We proceed to separate the background into a finance- and security-related overview.

A. Finance Background

Market Maker: Market makers (MM) help the market (i.e., the traders buying and selling assets) having access to sufficient liquidity (i.e., monetary asset amounts) for buy/sell orders to match at the ask/bid price. Traditionally, market makers are incentivized to operate as they can profit from the spread (i.e., the difference) between the bid and ask prices.

Automated Market Maker Exchanges: AMMs govern through smart contracts a pool of assets, where a pricing formula defines the asset purchase and sell price. Several AMM pricing formulas are proposed in the literature; the most popular form is the constant product AMM [35]. While Bancor introduced the AMM concept, at the time of writing, Uniswap [3] is the most prominent AMM with a daily trading volume of over $944M$ USD and $5.06B$ USD of the total supplied liquidity among $31,202$ different asset pairs[3]. One of the better-known forks of Uniswap is Sushiswap [2].

Arbitrage: The process of selling/buying an asset in one market while concurrently buying/selling in another market at a different price is known as arbitrage. Arbitrage encourages economic stability and is generally regarded as benign. DeFi traders/miners track new blockchain state changes and conduct arbitrage if the anticipated revenue from synchronizing two markets exceeds the expected transaction costs. To perform arbitrage, a trader may operate on the previous block state or on the state of the pool of unconfirmed transactions (i.e. the mempool) [46].

Slippage: The adjustment in the price of an asset during a transaction is known as price slippage. Expected price slippage is the anticipated rise or decrease in price depending on the amount to be exchanged and the available liquidity [53]. The expected slippage increases as trading volume increases. Unexpected price slippage is the rise or decrease in price during the interim time between creating a transaction and its execution. The sum of the expected and unexpected slippage represents the price impact of a trade.

Swap Routing Aggregators: An exchange aggregator is a service to aggregate liquidity from multiple exchanges. Aggregators may split a single trade into numerous smaller transactions to receive the best overall trade price. The sub-trades are then routed to various exchanges to provide the best exchange price and minimize the trading slippage. In March 2021, the three most significant off-chain aggregators (1inch, Mocha, and Paraswap) amassed a total monthly volume of $11.49B$ USD [4]. Off-chain aggregators are not guaranteed to yield optimal execution parameters due to the unexpected slippage. To the best of our knowledge, off-chain aggregators also do not perform arbitrage.

Flash Loans: Atomic blockchain transactions may execute several actions in a rigorous sequence. If a single transaction fails in one of its execution steps, the entire transaction fails atomically and does not alter the blockchain state. This atomicity property enables a novel financial product, flash loans. Flash loans are loans drawn from a smart contract pool of assets and are only valid within one atomic transaction. The flash loan must be paid back by the end of the transaction; otherwise, the loan fails. When a flash loan fails, the blockchain state is not modified, corresponding to a state where the loan was never granted to the borrower. Because lenders bear no risks by the borrowers defaulting on the loan, flash loans quickly grow to billions of USD [47].

B. Security Background

Front- and Back-running: is the process by which an adversary observes a victim's pending transaction on the network layer and then acts upon this information by placing trades before or after the victim's transaction. While custodian and centralized financial services are known to be under the supervision of regulatory bodies [4], DeFi (and blockchain) are not yet thoroughly regulated. Previous studies have observed front-running bidding wars between DEX arbitrage bots [22], [46]. Transaction fee bidding causes on-chain congestion and introduces network layer overhead, which necessarily increases the stale block rate and hence weakens the consensus security of the underlying blockchain [30].

Sandwich Attack: A sandwich attack is a predatory trading strategy, which exploits a pending, not yet executed trade [53]. Suppose an asset's price is expected to rise/fall due to a pending trade. In that case, a malicious front-runner can buy/sell the asset before the victim transaction executes and then close its position by selling/buying the same asset after the victim transaction is confirmed. Because AMMs provide complete transparency about the exchange's state and the pricing formula, sandwich attackers can derive the optimal attack parameters. Previous works investigate AMM-specific mitigations and find that sandwich attacks are not profitable if the victim's input amount remains below a safe, market-state-specific threshold.

[3] https://info.uniswap.org/

[4] https://www.theblockcrypto.com/data/decentralized-finance/dex-non-custodial/dex-aggregator-trade-volume

Miner Extractable Value: Miners retain the authority to decide on the transaction order of their mined blocks. Miners observe pending transactions on the network layer and may maximize their revenue through the optimal transaction order. For instance, the miners can perform front-running or sandwich attacks [53]. The term "Miner Extractable Value" or MEV [22], refers to the entire potential revenue that miners can extract through transaction order manipulation. Related work quantified that at least $28.8M$ USD in profit was extracted over two years following the 1st December 2018 [46]. Because non-MEV miners order by default transactions in descending transaction fee (gas price) amount [53], a non-mining trader can also capture MEV by adjusting their transaction fees. Related work, for example, shows how trading bots engage in competitive transaction fee bidding contests [46].

Stale Block Rate: Previous works have extensively shown that blockchain forks increase the stale block rate and deteriorate the consensus security by increasing the risks of double-spending and selfish mining [30], [28], [17]. Zhou *et al.* [52] quantified an MEV threshold at which MEV-aware miners are incentivized to fork the blockchain. For instance, on Ethereum, a miner with a hash rate of 10% would fork the blockchain if an MEV opportunity exceeds 4× the block reward. Because arbitrage is one of the prime sources of MEV, it is therefore of utmost importance to minimize the need for arbitrage to increase blockchain security.

C. AMM Arbitrage

We outline the traditional AMM design coupled with the necessary third-party arbitrageurs in Figure 3. This AMM design necessarily requires at least two separate transactions, TX_{swap} and TX_{arb} to synchronize the prices on AMM1 and AMM2 after the swap. Moreover, because TX_{swap} and TX_{arb} are non-atomic (i.e., do not necessarily execute in succession), multiple arbitrageurs, as well as miners, are likely to compete over benefiting from the created arbitrage opportunity [22].

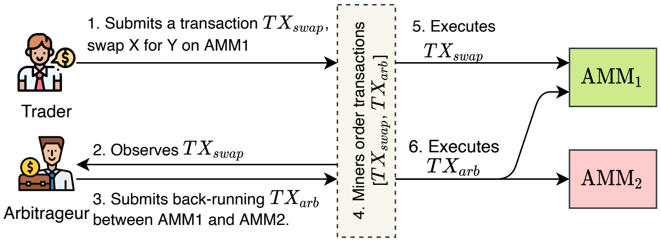
1. Submits a transaction TX_{swap}, swap X for Y on AMM1
Trader
2. Observes TX_{swap}
Arbitrageur
3. Submits back-running TX_{arb} between AMM1 and AMM2.
4. Miners order transactions [TX_{swap}, TX_{arb}]
5. Executes TX_{swap}
6. Executes TX_{arb}
AMM$_1$
AMM$_2$

Fig. 3: Overview of the back-running arbitrage process in AMM exchanges. The liquidity taker initiates a swap on AMM1 by broadcasting its transaction (TX_{swap}) on the P2P network. An arbitrageur listens on the public P2P network and observes TX_{swap}. The arbitrageur then issues a back-running arbitrage transaction (TX_{arb}), if TX_{swap} creates a profitable arbitrage opportunity between AMM1 and AMM2. Note that miners can collude with arbitrageurs to extract profits without failure risks.

D. Off-chain AMM routing

Off-chain swap routing services calculate the best routing path and parameters based on their local blockchain state (cf. Figure 4). Off-chain routing avoids complex smart contract operations, thereby minimizing transaction costs. However, off-chain routing paths and parameters are not necessarily optimal during execution, because the blockchain state might change intermittently between route generation and execution. To mitigate this problem, aggregators (such as 1inch) cooperate with miners on privately mined transactions, which arguably renders DeFi more centralized [6], [46]. Moreover, the goal of routing is to only optimize the liquidity takers' swap transactions, while forgoing possible arbitrage opportunities between other AMM exchanges.

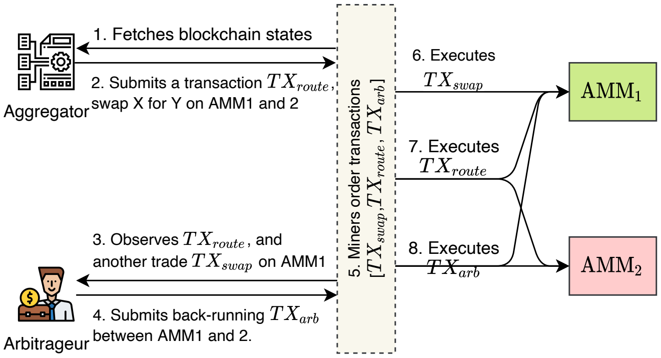
1. Fetches blockchain states
2. Submits a transaction TX_{route}, swap X for Y on AMM1 and 2
Aggregator
3. Observes TX_{route}, and another trade TX_{swap} on AMM1
4. Submits back-running TX_{arb} between AMM1 and 2.
Arbitrageur
5. Miners order transactions [TX_{swap}, TX_{route}, TX_{arb}]
6. Executes TX_{swap}
7. Executes TX_{route}
8. Executes TX_{arb}
AMM$_1$
AMM$_2$

Fig. 4: Off-chain routing aggregator. Upon (0) incoming swap, the routing service calculates (1) optimal paths and trading parameters given a blockchain state. The routing service (2) issues the transaction (tx_{route}) on the P2P network. An arbitrageur (3) observes tx_{route}, and another swap (tx_{swap}). The arbitrageur then (4) performs a back-running arbitrage (tx_{arb}). Because the transaction order execution is not guaranteed off-chain routing paths and parameters may be suboptimal.

III. A^{2}MM

We proceed to introduce the A^{2}MM design, system, threat, and state transition model by adding arbitrage actions on top of a known AMM model [53]. Note that we only study two-point arbitrages, and we, therefore, focus on markets with two assets in the following.

A. System Model

We consider a blockchain P2P network, where traders interact with AMMs by signing transactions with their respective private keys. For example, traders can exchange cryptocurrency assets, deposit/withdraw assets to/from different exchange pools, perform arbitrages, etc. Traders can freely adjust the parameters of these transactions, such as the transaction fees (e.g., gas price), slippage limit, expiration time, etc. A trader then broadcasts a transaction on the asynchronous blockchain peer-to-peer (P2P) network [23], [38], [37], or may privately send transactions to miners [53]. The transaction propagation typically utilizes gossip or publish-subscribe mechanisms, and nodes (including miners) may have different views of the pending (i.e., unconfirmed) transactions stored in the mempool. By default, miners order transactions according to the paid transaction fees but were also shown to adhere to private ordering policies [46].

B. Threat Model

We do not constrain the mining behavior of the miners but assume that no miner can accumulate more than 33% of the total hash-rate [28]. Miners can manipulate transaction ordering by transparently ignoring the default transaction ordering rules (i.e., highest-priced transactions first) or by attempting to hide private agreements by pretending to participate in transaction fee bidding contests [46]. We assume that smart contracts are secure and free from vulnerabilities.

C. AMM State Transition

We follow the standard model for AMM exchanges [53]. An AMM consists of mainly two types of **traders**, namely the **liquidity providers** and **liquidity takers**. A liquidity taker buys or sells an asset in exchange for another asset, using the liquidity providers' disposable assets.

AMM State

Definition III.1. The state (or depth) of an AMM market with two assets X and Y is defined as $(x, y) \in (\mathbb{R}^+)^2$. The sum of x and y correspond the total amount of assets from X and Y deposited by liquidity providers.

Two-asset AMMs typically support the following two-state transition functions for liquidity takers to convert between asset X and Y.

1) $\text{Swap}X\text{to}Y$: A liquidity taker can trade δ_x of asset X, increasing the available liquidity of asset X, in exchange for δ_y of asset Y, decreasing the available liquidity of asset Y (cf. Equation 1).

$$\begin{aligned} (x, y) &\xrightarrow[\delta_x \in \mathbb{R}^+]{\text{Swap}X\text{to}Y(\delta_x, \delta_y)} (x + \delta_x,\ y - \delta_y) \\ \text{where:} \quad & \delta_y \leq \Delta_y = p_{X \to Y}(x, y, \delta_x) \cdot \delta_x \end{aligned} \tag{1}$$

2) $\text{Swap}Y\text{to}X$: The mirroring action for $\text{Swap}X\text{to}Y$.

Note that both $\text{Swap}X\text{to}Y$ and $\text{Swap}Y\text{to}X$ use a pricing function $p(\cdot)$ to determine the maximum amount of asset Y the taker can receive. Each AMM exchange may choose a custom pricing function $p(\cdot)$ for governing the asset exchange. A liquidity taker can exchange δ_x amount of asset X for up to Δ_y amount of asset Y, while choosing to withdraw fewer assets Y voluntarily.

Pricing Formula

Definition III.2. A pricing formula is a differentiable function $p_{X \to Y}(x, y, \delta_x) : (\mathbb{R}^+)^3 \mapsto \mathbb{R}^+$, which maps the AMM state (x, y) and input amount (δ_x) of asset X to the best exchange rate the AMM offers.

Assumptions: We assume that the AMMs we consider abide by the following properties.

Liquidity Sensitivity

Property 1. Given an AMM state (x, y), the price $p_{X \to Y}(x, y, \delta_x)$ decreases as the trade size (δ_x) increases. Similarly, the price $p_{Y \to X}(x, y, \delta_y)$ decreases as the trade size (δ_y) increases.

Liquidity sensitivity (cf. Property 1) enables the underlying AMM market to adjust autonomously the price based on the trading volume and direction. The more asset X a liquidity taker purchases from an AMM, the more scarce X becomes in the liquidity pool, and therefore the price of X increases relative to Y (and vice versa). Property 1 implies that the pricing functions are monotonically decreasing as the trade volume increases.

Path Independence

Property 2. Given an inital market state (x, y), the following sub-properties holds:

1) Two consecutive $\text{Swap}X\text{to}Y$ transactions, respectively swapping δ_x^1, δ_x^2 asset X to δ_y^1, δ_y^2 asset Y, are equivalent to one $\text{Swap}X\text{to}Y$ transactions, swapping $\delta_x^1 + \delta_x^2$ asset X to $\delta_y^1 + \delta_y^2$ asset Y.
2) Two consecutive transactions, swapping δ_x^1 asset X to δ_y^1 asset Y($\text{Swap}X\text{to}Y$) followed with δ_y^2 asset Y back to δ_x^2 asset X($\text{Swap}Y\text{to}X$), where $\delta_x^1 - \delta_x^2 = \delta_x$, are equivalent to one $\text{Swap}X\text{to}Y$ transaction, swapping $\delta_x^1 - \delta_x^2$ asset X for $\delta_y^1 - \delta_y^2$ asset Y.

Path Independence is a desirable AMM property because it ensures that liquidity takers have no incentive to split a trade into multiple smaller transactions on the **same** AMM market. Note that when there exist numerous appropriate AMM markets, it can still be more profitable to split a trade and perform routing (cf. Section III-D).

Market Independence

Property 3. Given two state transition actions on two different AMM markets, the execution order of these two transitions will not impact the final states of the two AMMs.

Property 2 and 3 are applied in the following to compress routing and arbitrage transactions in Section III-G. Note that the same AMM can have multiple markets trading the same asset pairs (X, Y), but we assume these markets to have different states. A state change on one market will hence not affect the state or price of another market.

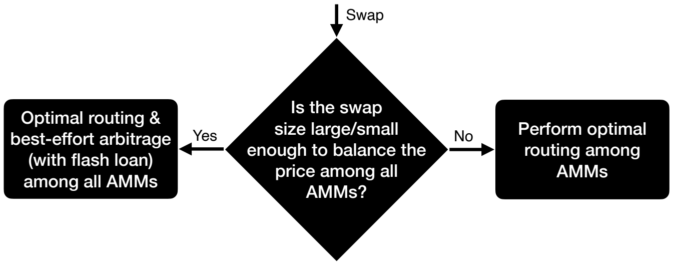
Swap
Is the swap size large/small enough to balance the price among all AMMs?
Yes
Optimal routing & best-effort arbitrage (with flash loan) among all AMMs
No
Perform optimal routing among AMMs

Fig. 5: Decision tree of the A^{2}MM exchange logic. A^{2}MM encourages first optimal routing, and if routing alone isn't sufficient, best-effort arbitrage may synchronize the markets.

D. A^{2}MM Design

In the following section we describe the proposed A^{2}MM design (cf. Figure 1). On a high-level, A^{2}MM minimizes the arbitrage opportunities between itself and other AMM exchanges, after any A^{2}MM state transition (e.g., a *swap*, *add* or *remove* liquidity). Upon an incoming swap, the A^{2}MM first checks if the swap amount is sufficiently large to synchronize the prices on the considered AMMs, and then performs optimal routing (cf. Figure 5). Otherwise, A^{2}MM performs optimal routing and best-effort arbitrage among the considered AMMs. A flash loan can be requested if the swap's trader holds an insufficient balance for arbitrage [47].

We assume that an A^{2}MM market (X, Y) is synchronizing with N other AMM X, Y markets. The state of the ith ($1 \le i \le N$) AMM X, Y market is denoted as (x_i, y_i). We formally introduce the state transitions for arbitrage and routing.

1) ArbitrageFor$X$: An arbitrageur can perform an arbitrage between two or multiple AMM exchanges. Given two AMMs with states (x_i, y_i) and (x_j, y_j) respectively. The trader initiates the arbitrage by swapping δ_x of X for δ_y amount of Y on AMM i. The arbitrageur then reverses the trade by exchanging δ_y amount of Y on AMM j for δ'_x amount of X. If the arbitrage is successful, the arbitrageur gains $\delta'_x - \delta_x$ amounts of X (cf. Equation 2).

$$\begin{aligned}
&(x_i, y_i, x_j, y_j) \xrightarrow[\delta_x\in\mathbb{R}^+]{\text{ArbitrageFor}X(\delta_x, i, j)} \\
&\qquad\qquad (x_i + \delta_x, y_i - \delta_y, x_j - \delta'_x, y_j + \delta_y) \\
&\iff \\
&(x_i, y_i) \xrightarrow[\delta_x\in\mathbb{R}^+]{\text{Swap}X\text{to}Y(\delta_x, \delta_y)} (x_i + \delta_x,\ y_i - \delta_y) \\
&(x_j, y_j) \xrightarrow[\delta_y\in\mathbb{R}^+]{\text{Swap}Y\text{to}X(\delta_y, \delta'_x)} (x_j - \delta'_x,\ y_j + \delta_y) \\
&\text{given } \delta'_x - \delta_x > 0, \text{s.t.} \begin{cases} \delta_y \le p_i^{X\to Y}(x_i, y_i, \delta_x)\cdot\delta_x \\ \delta'_x \le p_j^{Y\to X}(x_j, y_j, \delta_y)\cdot\delta_y \end{cases}
\end{aligned} \tag{2}$$

2) ArbitrageFor$Y$: The mirroring action for ArbitrageFor$X$.
3) Route$X$to$Y$: When a liquidity taker swaps δ_x of asset X for δ_y of asset Y, the taker can split its trade $(\delta_{x1}, \dots, \delta_{xN})$ across N AMMs (cf. Equation 3).
4) Route$Y$to$X$: The mirroring action for Route$Y$to$X$.

$$\begin{aligned}
&(x_1, y_1, \dots, x_N, y_N) \xrightarrow[\delta_{xi}\in\mathbb{R}^+,\quad \forall 1\le i\le N]{\text{Route}X\text{to}Y(\delta_x)} \\
&\qquad\qquad (x_1 + \delta_{x1}, y_1 - \delta_{y_1}, \dots, y_N - \delta_{y_N}) \\
&\iff \\
&(x_i, y_i) \xrightarrow[\delta_{xi}\in\mathbb{R}^+, \forall 1\le i\le N]{\text{Swap}X\text{to}Y(\delta_{xi}, \delta_{yi})} (x_i + \delta_{xi},\ y_i - \delta_{yi}) \\
&\qquad \text{s.t.} \begin{cases} \delta_x = \sum_{i=1}^N \delta_{xi}, \quad \delta_y = \sum_{i=1}^N \delta_{yi} \\ \delta_{yi} \le p_i^{X\to Y}(x_i, y_i, \delta_{xi})\cdot\delta_{xi} \end{cases}
\end{aligned} \tag{3}$$

E. Optimal On-Chain Swap Routing

A^{2}MM performs Route$X$to$Y$ to maximize the amount of asset Y the liquidity taker purchases after paying δ_x. We proceed to formally define the optimal routing problem among N AMMs.

$$\begin{aligned}
&\text{maximize } \delta_y = \sum_{i=1}^N \delta_{yi} \quad \text{given } (x_i, y_i)\ \forall\ 1 \le i \le N \\
&\text{s.t. } (x_1, y_1, \dots, x_N, y_N) \xrightarrow[\delta_{xi}\in\mathbb{R}^+,\ \forall 1\le i\le N]{\text{Route}X\text{to}Y(\delta_x), \delta_x = \sum_{i=1}^N \delta_{xi}} \\
&\quad (x_1 + \delta_{x1}, x_1 - \delta_{y_1}, \dots, x_N + \delta_{x_N}, y_N - \delta_{y_N}),
\end{aligned} \tag{4}$$

Theorem 1. *Routing optimization aims to level the asset price on multiple AMMs and can be solved by greedily routing transaction volume.*

PROOF BY CONTRADICTION: This proof shows that the optimal routing among N AMMs must greedily route transaction volume to the exchange with the best price (cf. Theorem 1). We assume the existence of an optimal routing strategy (S_{optimal}) for Route$X$to$Y$, which in total routes δ_x amount of asset X to δ_y amount of asset Y. More specifically, this optimal strategy routes $\delta_{x1}, \dots, \delta_{xN}$ of asset X to N AMMs, in exchange of $\delta_{y1}, \dots, \delta_{yN}$ of asset Y ($\delta_x = \sum_{z=1}^N \delta_{xz}$, $\delta_y = \sum_{z=1}^N \delta_{yz}$). After the routing, we assume that AMM 2 still offers a better price than AMM 1, meaning that S_{optimal} contradicts Theorem 1 and does not route all trading volume greedily to the AMM with the best price. Equation 5 shows the state transition process.

$$\begin{aligned}
&(x_1, y_1) \xrightarrow{\text{Swap}X\text{to}Y(\delta_{x1}, \delta_{y1})} (x'_1, y'_1) \\
&(x_2, y_2) \xrightarrow{\text{Swap}X\text{to}Y(\delta_{x2}, \delta_{y2})} (x'_2, y'_2) \\
&\delta_y = p_{X\to Y}(x_1, y_1, \delta_{x1})\delta_{x1} \\
&\qquad + p_{X\to Y}(x_2, y_2, \delta_{x2})\delta_{x2} + \sum_{z=3}^N \delta_{yz}
\end{aligned} \tag{5}$$

To prove that S_{optimal} is not the optimal routing strategy, we show that the routing can output more asset Y if more trading volume is routed to AMM 2, without changing the routing strategy for AMM 3 to N. We denote this alternative strategy as S_{alter}, which routes $\delta_{x1} - \Delta, \delta_{x2} + \Delta$ of asset X to AMMs 1 and 2 respectively. AMM 2 still offers a better price for swap$X$to$Y$ after executing S_{alter}, because the additionally routed amount

(Δ) is arbitrarily small and $P_{X\to Y}$ is a continuous function. Equation 6 shows the state transition process for S_{alter}.

$$\begin{aligned}
&(x_1, y_1) \xrightarrow{\text{Swap}X\text{to}Y(\delta_{x1}-\Delta,\delta_{y1}*)} (x_1*, y_1*)\\
&(x_2, y_2) \xrightarrow{\text{Swap}X\text{to}Y(\delta_{x2}+\Delta,\delta_{y2}*)} (x_2*, y_2*)\\
&\delta'_y = p_{X\to Y}(x_1, y_1, \delta_{x1} - \Delta)\cdot(\delta_{x1} - \Delta)\\
&\qquad + p_{X\to Y}(x_2, y_2, \delta_{x2} + \Delta)\cdot(\delta_{x2} + \Delta) + \sum_{z=3}^{N}\delta_{yz}\\
&\text{where: } p_{X\to Y}(x_1*, y_1*, \Delta) < p_{X\to Y}(x_2*, y_2*, \Delta)
\end{aligned} \tag{6}$$

Based on the liquidity sensitivity property (Property 1), AMM 2 offers a worse price for Swap$X$to$Y$ after executing S_{alter} compared to executing S_{optimal} (cf. Equation 7). This is because S_{alter} routes more trade volume to AMM 2, where both strategies have the same initial state for AMM 2 (x_2, y_2).

$$p_{X\to Y}(x_2*, y_2*, \Delta) < p_{X\to Y}(x'_2, y'_2, \Delta) \tag{7}$$

In Equation 8 we derive that the amount of asset Y S_{alter} outputs is greater than S_{optimal} using the path independence property (cf. Property 2), which contradicts the assumption that S_{optimal} is the optimal routing strategy. Theorem 1 is therefore proven by contradiction. □

$$\begin{aligned}
\delta'_y - \delta_y =\ & p_{X\to Y}(x_1, y_1, \delta_{x1} - \Delta)\cdot(\delta_{x1}-\Delta)\\
& + p_{X\to Y}(x_2, y_2, \delta_{x2} + \Delta)\cdot(\delta_{x2}+\Delta)\\
& - p_{X\to Y}(x_1, y_1, \delta_{x1})\cdot\delta_{x1} - p_{X\to Y}(x_2, y_2, \delta_{x2})\cdot\delta_{x2}\\
=\ & \cancel{p_{X\to Y}(x_1, y_1, \delta_{x1}-\Delta)\cdot(\delta_{x1}-\Delta)} + \cancel{p_{X\to Y}(x_2, y_2, \delta_{x2})\cdot\delta_{x2}}\\
& + p_{X\to Y}(x'_2, y'_2, \Delta)\cdot\Delta\\
& - \cancel{p_{X\to Y}(x_1, y_1, \delta_{x1}-\Delta)\cdot(\delta_{x1}-\Delta)} - \cancel{p_{X\to Y}(x_2, y_2, \delta_{x2})\cdot\delta_{x2}}\\
& - p_{X\to Y}(x*_1, y*_1, \Delta)\cdot\Delta\\
=\ & p_{X\to Y}(x'_2, y'_2, \Delta)\cdot\Delta - p_{X\to Y}(x_1*, y_1*, \Delta)\cdot\Delta\\
\geq\ & p_{X\to Y}(x'_2, y'_2, \Delta)\cdot\Delta - p_{X\to Y}(x_2*, y_2*, \Delta)\cdot\Delta \geq 0
\end{aligned} \tag{8}$$

F. Arbitrage Profit Maximization

In the following, we formally introduce the arbitrage profit maximization problem between N AMMs. An arbitrage between multiple DEXes may include multiple sub-arbitrage steps. Given an arbitrage strategy with L steps, we use the superscript, such as x^l, y^l, to denote the state and parameters at a sub-step l, where $1 \leq l \leq L$. The objective of the arbitrageur is to maximize the revenue after executing all sub-arbitrage steps (cf. Equation 9). Because the solution of Equation 9 depends on the implementation-specific AMM pricing formulas for the N AMMs, we do not provide a general solution here. The reader, however, can find an optimal solution for two AMMs with constant product pricing formula in Section B-B (corresponding to Uni- and Sushiswap capturing over 73.27% of the total AMM market trading volume at the time of writing[5], accessed March 2021).

$$\begin{aligned}
&\text{given: } (x_i, y_i)\ \forall\ 1 \leq i \leq N\\
&\text{maximize: } \sum_{l=0}^{L}\delta'^{l}_x - \delta^l_x\\
&\text{subject to: for each sub-step } l \text{ (in total } L \text{ steps) :}\\
&\qquad\begin{cases}
(x^{l+1}_{i_l}, y^{l+1}_{i_l}, x^{l+1}_{j_l}, y^{l+1}_{j_l})\\
\qquad = \text{ArbitrageFor}X(\delta^l_x, i_l, j_l)\\
\qquad = (x^l_{i_l} + \delta^l_x, y^l_{i_l} - \delta^l_y, x^l_{j_l} - \delta'^{l}_x, y^l_{j^l} + \delta^l_y)\\
(x^{l+1}_k, y^{l+1}_k) = (x^l_k, y^l_k), \quad \text{if } k \notin \{i_l, j_l\}
\end{cases}\\
&\qquad 1 \leq i^l \leq N, 1 \leq j^l \leq N, i^l \neq j^l, \quad \forall 1 \leq l \leq L
\end{aligned} \tag{9}$$

G. Swap Compression

To minimize the required transaction fees of a swap, we show in the following how A^{2}MM compresses multiple transactions of an atomic swap (cf. Figure 6).

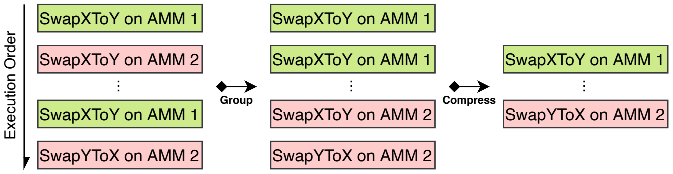

Fig. 6: Overview of the swap compression process.

Theorem 2. *An optimal strategy ($S_{optimal}$) performing routing and arbitrage among N AMMs on market (X, Y) is equivalent to a batch execution strategy (S_{batch}). S_{batch} consists of at most N swaps (Swap$X$to$Y$ or Swap$Y$to$X$). Both $S_{optimal}$ and S_{batch} change the states of AMMs from $(x_1, y_1, \ldots, x_N, y_N)$ to $(x'_1, y'_1, \ldots, x'_N, y'_N)$.*

PROOF: Both arbitrage and routing only consist of Swap$X$to$Y$ and Swap$Y$to$X$ transactions (cf. Equation 2 and 3). Because the AMMs are independent (cf. Property 3), these transactions can be reordered into N groups, where each group only consists of transactions for the same market. We can then batch the transactions within each group based on the path independence property (Property 2). □

H. Limitations

In this work, we consider A^{2}MM in isolation from other exchanges on other blockchains or external centralized exchanges. However, asset prices realistically move outside of the regarded AMM, which may still create arbitrage opportunities, even if A^{2}MM minimizes required arbitrages among the synchronized AMMs. Moreover, the cost of price synchronization grows with the number of AMMs that A^{2}MM peers with and is therefore limited (cf. Section VI). While in this work we only consider AMM with similar pricing formulas, we believe that A^{2}MM is adoptable any AMM pricing formula, given the corresponding on-chain computation overhead.

[5] https://www.theblockcrypto.com/data/decentralized-finance/dex-non-custodial/dex-volume-monthly

IV. Evaluation

In our evaluation we rely on the blockchain states of Uni- and Sushiswap, two of the biggest on-chain DEXes capturing 73.27% of the market volume at the time of writing[6]. Therefore, our A^{2}MM implementation peers with two AMMs of the same pricing formula. In this scenario, the optimal strategy for both routing and arbitrage can be mathematically derived as we show in the appendix (cf. Section B). We use Uniswap as a pricing oracle to fetch the X/ETH prices for any arbitrary asset X[7]. We assume that the X/ETH price is zero when we cannot determine the price, thus ignoring the corresponding transaction. We adopt a price of 2000 USD/ETH as of April 2021.

A. Empirical Comparison of AMM and A^{2}MM

We perform an empirical comparison between AMM and A^{2}MM through concrete execution on past blockchain data. Our experimental setup corresponds to the system model in Figure 7, where we assume that we deploy an A^{2}MM contract with a user interface while using Uni- and Sushiswap's liquidity pools. Upon receiving a swap request, A^{2}MM derives the routing and arbitrage parameters on the fly on-chain.

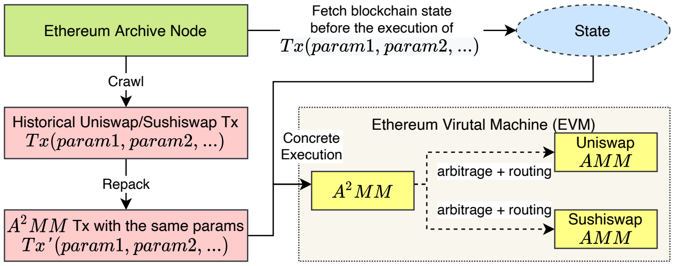
Ethereum Archive Node
Fetch blockchain state before the execution of $Tx(param1, param2, ...)$
State
Crawl
Historical Uniswap/Sushiswap Tx
$Tx(param1, param2, ...)$
Repack
A^2MM Tx with the same params
$Tx'(param1, param2, ...)$
Ethereum Virutal Machine (EVM)
Concrete Execution
A^2MM
arbitrage + routing
Uniswap AMM
arbitrage + routing
Sushiswap AMM

Fig. 7: System model of the A^{2}MM evaluation. We concretely execute on past blockchain data, and assume that A^{2}MM initially does not provide a liquidity pool and rather operates through Uni- and Sushiswap.

We implement A^{2}MM in 761 lines of code using Solidity v.8.2.0. The deployment of the A^{2}MM contract costs $2,821,822$ gas (0.282 ETH, 564 USD) at a gas price of 100 gWei. We crawl all asset swap transactions that are sent directly to Uni- or Sushiswap from block 10794261 ($4th$ September, 2020) to block 12000000 ($8th$ March, 2021) (185 days). Note that A^{2}MM aims to reduce the number of two-point arbitrage and sandwich opportunities, where arbitrage and sandwich bots often use smart contract accounts. Because we assume that all AMM swaps initiate at the A^{2}MM's user interface, we expect the number MEV related transactions to decrease. To avoid double-counting MEV transactions in our evaluation, we chose to only consider AMM swaps from non-smart contract accounts (i.e., EOA accounts).

Transaction Fees When Peered With Two AMMs: A^{2}MM requires more computation for a single swap than a standard AMM, because A^{2}MM derives the routing and arbitrage parameters across AMMs on-chain. A natural question is how much more expensive A^{2}MM's execution ends up when compared to an AMM, when it peers with Uni-/Sushiswap. Table I presents our concrete execution results. For a swap without arbitrage nor routing, we find that on average, liquidity takers pay 6.97% higher fees for a swap on A^{2}MM vs. AMM. For a swap with routing, we find that A^{2}MM requires an excess 17.80% in terms of transaction fees compared to the average transaction fee of an AMM swap. Finally, for a swap with routing and arbitrage, A^{2}MM's excess in transaction fees amounts to 60.22%. We estimate that the arbitrage action by itself costs $60.22\% - 17.80\% = 42.42\%$.

Extractable Arbitrage/Routing Revenue of A^{2}MM: In the following, we quantify the income potential from A^{2}MM's design, as arbitrage is known to yield positive incoming from synchronizing prices. Routing provides better swap asset prices by sourcing several liquidity pools simultaneously. We proceed to measure both the positive income from arbitrage and the price advantage from routing, allowing us to offer an objective view of the costs of using A^{2}MM compared to an AMM.

Our results suggest (cf. Table II) that within 185 days of blockchain data, in total $460,349$ (81.87%) of the executed A^{2}MM transactions perform either arbitrage and/or routing, extracting a total of $10,675$ ETH ($21,350,565$ USD). Due to this positive income and the routing price advantage, in expectation, A^{2}MM reduces transaction fees by an average of 90% compared to a standard AMM swap.

B. Two-point Arbitrage Overhead

While A^{2}MM only mitigates two-point arbitrage overhead, Qin *et al.* [46] show that historically 41% of the on-chain arbitrages are two-point arbitrages. Therefore, we estimate that A^{2}MM will decrease about 41% of the on-chain and network overhead caused by arbitrage bots, helping to reduce the stale block rate, thus increasing blockchain consensus security.

In the following, we quantify both the on-chain and network layer overhead for the past two-point arbitrages between Uni- and Sushiswap to test the above intuition.

Block-space Overhead Heuristics: In the following we use B_i to denote a block with height i, and tx_i^k to denote a transaction mined within block b_i at index k. We use $f_S(B_i) \mapsto S$ to denote the blockchain state after executing all transactions in block B_i. We use the function $f_S(B_i, tx1, \ldots, txN) \mapsto S$ to denote the blockchain state after iteratively applying transactions $tx1, \ldots, txN$ in the exact order on the blockchain state $S(B_i)$. In other words, if there are l transactions in block B_i, then $S(B_i) = S(B_{i-1}, tx_i^0, \ldots, tx_i^l)$. We use the function $f_A(s \in S, tx \in TX) \mapsto bool$ to classify whether a transaction tx successfully performs an arbitrage at blockchain state s (cf. Section E in Appendix).

We classify transaction tx_i^k as an block-space overhead caused by front-/back-running arbitrages if Heuristic C1, and one of Heuristics C2a and C2b are satisfied.

[6] https://www.theblockcrypto.com/data/decentralized-finance/dex-non-custodial/dex-volume-monthly, accessed March 2021

[7] To avoid overestimating the revenue, we estimate the ETH value of x amount of asset X by simulating a Uni/Sushiswap trade, instead of relying on the spot price.

Num. sub-swaps per swap	Total		1		2		> 2	
Transaction fee	AMM	A^{2}MM	AMM	A^{2}MM	AMM	A^{2}MM	AMM	A^{2}MM
1. Swap	127±32K	136±40K	121±24K	128±27K	182±39K	208±53K	258±49K	310±74K
2. Swap + Routing	-	164±60K	-	145±41K	-	225±62K	-	353±82K
3. Swap + Arbitrage	378±96K	228±68K	345±73K	203±49K	450±96K	282±69K	574±103K	390±82K

TABLE I: Empirical transaction fee comparison between the AMM and A^{2}MM model. We consider three cases: *(i)* without routing/arbitrage *(ii)* with routing, and *(iii)* with arbitrage.

Type	Num. of Txs(%)	Total Revenue	Avg. Revenue	Avg. A^{2}MM Fee	Avg. $\frac{\text{Revenue}}{\text{Fee}_{\text{A}^2\text{MM}}}$	Avg. $\frac{\text{Revenue}-\text{Excess Fee}}{\text{Fee}_{\text{AMM}}}$
ETH→Tokens	860.9K(53%)	12.0KETH(24MUSD)	0.01ETH(28USD)	0.02ETH(40USD)	70%	62%
Tokens→Tokens	494.2K(30%)	17.7KETH(35MUSD)	0.04ETH(72USD)	0.03ETH(56USD)	127%	137%
Tokens→ETH	271.5K(17%)	3.0KETH(6MUSD)	0.01ETH(22USD)	0.02ETH(32USD)	69%	60%
Total	1,626.6K(100%)	32.7KETH(65MUSD)	0.02ETH(40USD)	0.02ETH(43USD)	93%	90%
Routing	588.0K(36%)	5.2KETH(10MUSD)	0.01 ETH(18 USD)	0.02ETH(36USD)	48%	39%
Arbitrage	561.7K(35%)	27.5KETH(55MUSD)	0.05 ETH(98 USD)	0.03ETH(60USD)	164%	203%

TABLE II: Revenue and cost of our concrete execution on past blockchain data, replaying previous Uni/Sushiswap transactions on A^{2}MM (i.e., the δ_x for $\text{Swap}X\text{to}Y$ remains unchanged, cf. Equation 1). We adopt a price of 2000 USD/ETH as of April 2021. Excess fee is the additional transaction fee A^{2}MM costs when compared with an AMM (i.e., $\text{Fee}_{\text{A}^2\text{MM}}-\text{Fee}_{\text{AMM}}$).

Percentile	10	20	30	40	50	60	70	80	90
Profit(USD)	-20.3	-9.9	-3.8	-1.8	-1.0	-0.4	0.4	4.3	31.0

TABLE III: Percentile analysis of individual swaps' profit from our prototype implementation. We assume that traders pay the same transaction fee price (gas price) in our concrete execution. While only 37% of the swaps realize a positive profit, A^{2}MM in expectation provides a better price than an AMM.

Heuristic C1: If transaction tx_i^k performs a successful arbitrage, then tx_i^k is not classified as an block-space overhead (cf. Equation 10).

$$f_A(S(B_{i-1}, tx_i^0, \ldots, tx_i^{k-1}), tx_i^k) = \text{false} \qquad (10)$$

Heuristic C2a (Front-running): We test, if a re-positioning of tx_i^k as the first transaction in each of the previous five blocks (i.e. a 1-minute time window), would make tx_i^k a successful arbitrage transaction. This test allows us to classify whether tx_i^k is a failed front-running arbitrage overhead (cf. Equation 11).

$$f_A(S(b_j), tx_i^k) = \text{true}, \text{where}(i-5) \leq j \leq (i-1) \quad (11)$$

Heuristic C2b (Back-running): By iterating backwards through the transactions of the last 5 blocks, starting at tx_i^k's position, we sequentially interleave tx_i^k after each tx_j^l, and test through concrete execution, whether tx_i^k yields an arbitrage profit. This test allows us to identify whether a transaction attempted an arbitrage operation (cf. Equation 12).

$$\begin{gathered} f_A(S(b_j), tx_j^l, tx_i^k) = \text{true} \\ \text{where: } (i-5) \leq j \leq (i-1), \qquad tx_j^l \neq tx_i^k. \\ \text{gas price of } tx_i^k \leq \text{gas price of } tx_j^l \end{gathered} \qquad (12)$$

Network Layer Overhead Heuristics: We classify a transaction tx as a network layer overhead targeting a successful arbitrage transaction tx_{arb}, if the following three heuristics (N1, N2 and N3) are satisfied. Note that while a transaction may propagate on the P2P network, that transaction doesn't necessarily appear in the blockchain. The helper function $f_{\text{P2P}}(x \in TX, B)$ returns the time of a transaction or block's first known appearance on the P2P network.

Heuristic N1: N1 tests whether a transaction on the P2P layer is a failed arbitrage attempt. Given an identified successful on-chain arbitrage transaction tx_{arb}, we replace tx_{arb} iteratively with each transaction tx recorded by our P2P network node. If tx yields an arbitrage profit under concrete execution, we classify tx as a failed network layer arbitrage attempt, caused by either GPA or BRF. Heuristics N2 and N3 attempt to narrow down the issuance time of an arbitrage transaction tx. If both the two tests in Heuristics N2 and N3 are satisfied, we then classify tx as a network layer overhead transaction targeting tx_{arb}.

Heuristic N2: N2 tests the lower time of appearance of an arbitrage attempt. We find the earliest appearance of all tx_{arb} related transactions on the P2P network, such as: *(i)* the transaction that tx_{arb} attempts to front-/back-run (denoted as $tx_{tx_{arb}}^{\text{victim}}$), and *(ii)* all failed block-space overhead transactions competing with tx_{arb} (denoted as $TX_{tx_{arb}}^{\text{overhead}}$). For each network layer overhead transaction tx, we test if tx is discovered after the earliest appearance of all tx_{arb} related transactions on the P2P network (cf. Equation 13).

$$\begin{gathered} t_{tx_{arb}} = f_{\text{P2P}}(tx) < min(\{f_{\text{P2P}}(x)\}), \text{where} \\ x \in tx_{arb} \cup TX_{tx_{arb}}^{\text{overhead}} \cup tx_{tx_{arb}}^{\text{victim}} \cup tx_{tx_{arb}}^{\text{target}} \end{gathered} \qquad (13)$$

Heuristic N3: N3 tests the upper time of appearance of an arbitrage attempt. For each network layer overhead transaction tx, we test if tx is discovered before tx_{arb} is

Position	Front-running	Back-running	Total
Same block	13,460(73%)	117,728(90%)	131,188(88%)
After 1 block	2,876(16%)	7,657(6%)	10,533(7%)
After 2 blocks	909(5%)	2,118(2%)	3,027(2%)
After 3 blocks	465(3%)	1,353(1%)	1,818(1%)
After 4 blocks	419(2%)	860(1%)	1,279(1%)
After 5 blocks	268(1%)	775(1%)	1,043(1%)

TABLE IV: Statistics of the block-space overhead we detect. 95% of the on-chain failed arbitrages are mined within 1 block after the MEV arbitrage opportunity is extracted.

mined (cf. Equation 14).

$$f_{\text{P2P}}(tx) > f_{\text{P2P}}(b), \text{where } tx_{arb} \text{ is mined in } b \tag{14}$$

Empirical Results: To quantify the amount of P2P network layer overhead caused by two-point arbitrages, we modify the Ethereum geth client to store all transactions received on the P2P network layer over $213,538$ blocks (36 days) from block $11,813,201$ (Feb-08-2021) to block $12,055,081$ (Mar-17-2021). Intuitively, the number of transactions the Ethereum node can observe increases with the number of peer connections, the network bandwidth, and the computation power of the machine. Our geth client operates on a Ubuntu 20.04.1 LTS machine with AMD Ryzen Threadripper $3990X$ (64-core, 2.9 GHz), 256 GB of RAM and 4×2 TB NVMe SSD in Raid 0 configuration. We limit the geth client to at most $1,000$ connections with other Ethereum peers instead of the default of 50 peers (cf. Figure 8). We captured in total 246B transaction propagation messages from $81,736$ unique peers originating from $63,744$ unique IP addresses and $2,859,833$ unique IP:Port combinations.

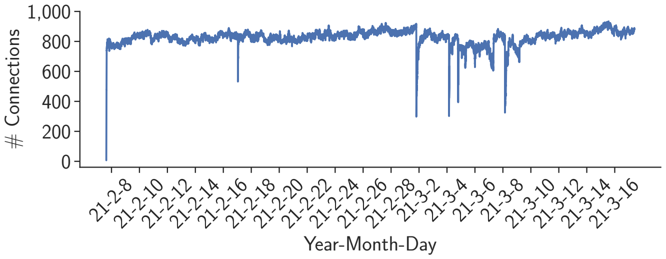

Fig. 8: Number of connections of our modified geth node while listening on the Ethereum P2P network from block $11,813,201$ (Feb-08-2021) to block $12,055,081$ (Mar-17-2021).

Potential Freed up Block-Space by A^{2}MM Given the heuristics C1 and C2, we identify in total $148,888$ on-chain two-point arbitrage overhead transactions (cf. Table IV). Surprisingly, the majority are back-running arbitrage failures ($130,491/87.64\%$). On average, 2.5 and 6.0 overhead transactions are mined on-chain for each front- and back-running opportunities, with an average gas cost of 193 ± 70K. As front-running arbitrageurs participate in PGA, front-runners pay a $1.56\times$ premium average gas price compared to back-runners. We use Equation 15 to quantify A^{2}MM's on-chain reduction over $213,538$ blocks (36 days), where C denotes the average on-chain space cost. We find that in expectation, A^{2}MM reduces the consumed block-space by 32.8%.

$$C_{\text{reduced}} = 1 - \frac{C_{\text{A}^2\text{MM swap}}}{C_{\text{AMM swap}} + C_{\text{arbitrage}} + C_{\text{block-space overhead}}} \tag{15}$$

Potential Network Overhead Reduced by A^{2}MM Given the heuristics N1, N2 and N3, we identify $105,960$ network overhead transactions, where the majority ($400,471/89.4\%$) are caused by back-running arbitrageurs. On average, 10.5 and 27.4 network overhead transactions are issued for every front-/back-running arbitrage opportunity, which corresponds to a factor of $4.2\times, 4.6\times$ more than the block-space overhead. When an arbitrage opportunity appears, we measure that the off-chain overhead transactions sum to an average of 5.83 ± 6.57 kb per block, which is around 13.8% of a block's size on the 1st of March 2021[8].

Limitations: Our evaluation may consist of false negatives. For example, an overhead arbitrage transaction may be dropped in the asynchronous P2P network before reaching our network node. The blockchain overhead statistics we report should therefore only be regarded as a lower bound of the actual network overhead. Note that a transaction is only classified as an overhead, if it does perform a two-point arbitrage during concrete execution. Therefore, our overhead evaluation suffers from no false positives.

V. Security Implications of A^{2}MM

In the following, we quantitatively outline the relevant security improvements A^{2}MM provides on the blockchain consensus.

A. *Stale Block Rate Simulation*

This section simulates the P2P network of four blockchains (Ethereum, Bitcoin, Litecoin, and Dogecoin) to estimate quantitatively the relationship between the stale block rate and the miner bandwidth. To capture the block propagation in the P2P network, we extend our system model from Section III. The asynchronous nature of blockchain P2P propagation is extensively studied by related works [26], [23], [30], [29], [53], on which we build upon.

Various factors influence block propagation, including the number of miners, the network topology, the peer internet latency, bandwidth, and overall network congestion. To ease our experiments and operate under the best network connectivity, we assume that the miners create direct point-to-point relations among themselves. Consequently, the number of sporadic network nodes, the network topology, intermediate devices (relay nodes, switches, and routers), and the TCP congestion management are all abstracted. We approximate the block propagation duration by dividing the block size over the bandwidth and adding the communication latency. To parameterize

[8] https://etherscan.io/chart/blocksize

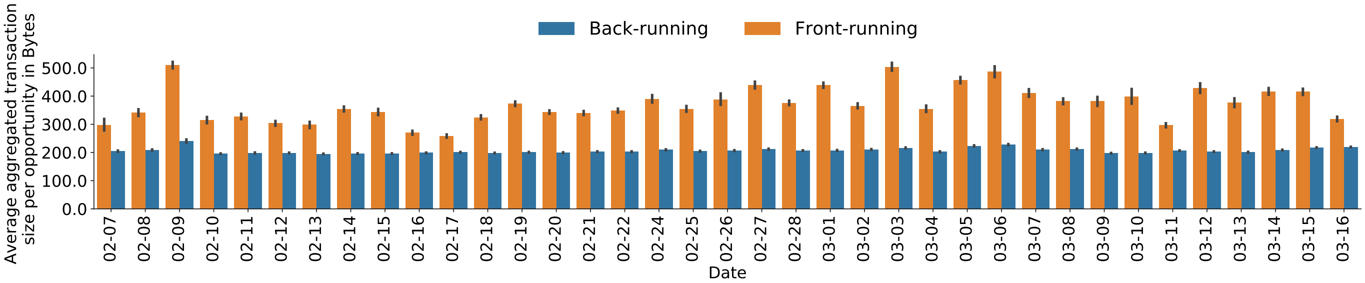

Fig. 9: For every two-point arbitrage opportunity on-chain, we correlate the accumulative overhead transactions propagated on the P2P network layer. We capture front- as well as back-running transactions, covering PGA and BRF.

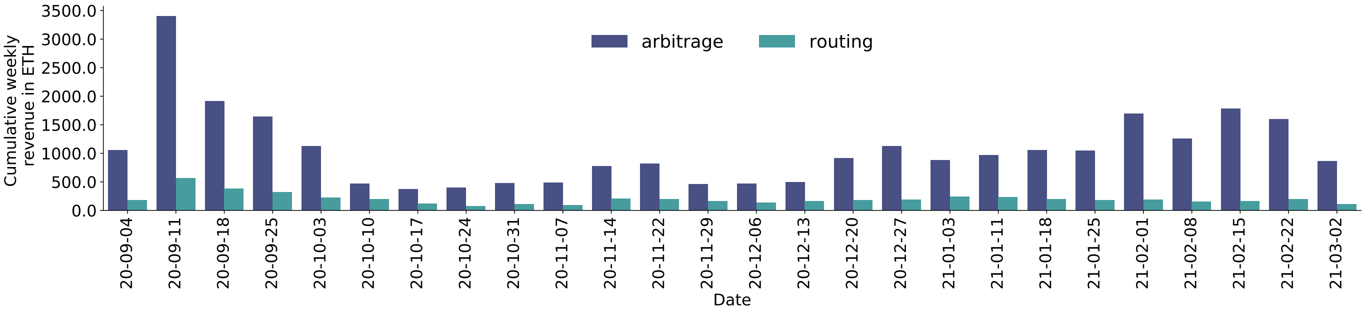

Fig. 10: Cumulative weekly revenue of the A^{2}MM implementation extracted through arbitrage and routing based on concrete execution of past blockchain data.

	Ethereum	Bitcoin	Litecoin	Dogecoin
Block interval (min)	0.223	9.474	2.59	1.07
Block size mean (kB)	44.0	863.8	61.1	15.9
Block size std (kB)	3.0	25.0	33.4	14.9

TABLE V: Blockchain parameters we use to simulate the P2P network for Ethereum, Bitcoin, Litecoin and Dogecoin. These statistics are measured using 90 days of blockchain data, from Jan-09-2021 to Apr-09-2021.

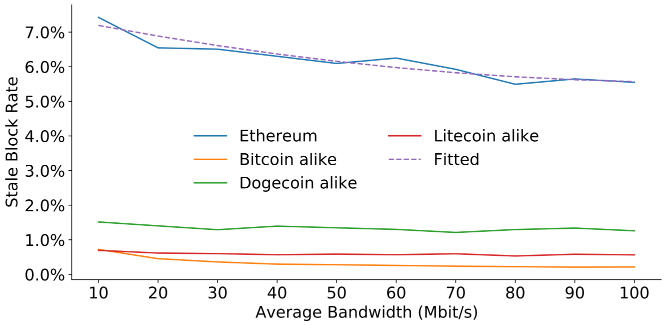

Fig. 11: Simulated stale block rate given the average P2P network bandwidth for Ethereum, Bitcoin, Litecoin and Dogecoin. We fit a least square regression line for Ethereum ($0.000158 \times \text{bandwidth}^2 - 0.03541 \times \text{bandwidth} + 7.531$).

a realistic block size distribution in our simulations, we assume that the block size follows a normal distribution, where the mean and variance are derived using 90 days of blockchain data (cf. Table V) [9]. To capture latency distribution, we apply the mean percentile statistics [38], [29], [53] and use linear interpolation to estimate the underlying cumulative probability distribution (cf. Table IX in the Appendix). We only consider the hashing power of the top 10 miners for each blockchain (cf. Table VIII in the Appendix), and assume that miners have a symmetrical upload and download bandwidth.

MEV network overhead deteriorates the miners' P2P bandwidth and hence increases the blockchain's stale block rate. The most significant arbitrage back-running in terms of overhead we capture amounts to a total of 1.92 Mb data from repeated transactions within one block interval (13 seconds). Note that the total amount of data miners receive is amplified as the number of connected peers increases. For instance, if miners have an initial upload and download speed of 70Mbit/s, and overhead transactions are propagated back and forth 200 times, the average bandwidth will decrease to 40.5Mbit/s [10]. Based on our 2nd degree least-square polynomial fitting, this decrease in bandwidth leads to a 0.53% increase in stale block rate (cf. Figure 11).

Note that miners have an incentive to connect with as many nodes as possible to minimize the risks of eclipse attacks [33], while the network layer overhead is amplified with the number of connections. We leave the estimation of such network

[9] https://bitinfocharts.com, accessed Apr 2021

[10] $\frac{1.92\text{Mb}}{13\text{s}} \times 200 = 40.5\text{Mbit/s}$

	Front-running	**Back-running**
Strategy	Priority Gas Auctions	Back-run Flooding
Target State	Confirmed Block State	Pending Block State
	Overhead	
On-chain?	Yes	Yes
Network?	Yes	Yes

TABLE VI: If an MEV bot acts on a confirmed block state, it performs PGA against other competing front-running bots. If an MEV bot acts on a pending block state, we observe back-run flooding. Both GPA and BRF cause on-chain and network layer overheads.

overhead amplification factor to future works.

B. Sandwich Attack Mitigation

Sandwich attacks are not profitable if the victim's input amount remains below the MVI [53]. This threshold depends on the AMM pricing formula, the total underlying pool liquidity, as well as the trader's slippage configuration. The MVI threshold for instance increases if the market liquidity increases.

By routing the trading volume onto multiple AMM exchanges, A^{2}MM aggregates the MVI thresholds among the underlying liquidity pools. In the simple case, where two AMM markets have the same liquidity and pricing formula, A^{2}MM's accumulative MVI threshold is $2\times$ the MVI of a single AMM.

C. Back-run Flooding Overhead Reduction

We observe *back-run flooding* on the P2P network, where MEV bots broadcast multiple similar back-running transactions for a single MEV opportunity (cf. Table VI). It appears that BRF may increase the success rate of back-running. For instance, each of the flooding transactions is likely to follow a different network propagation path in the asynchronous P2P network, which could increase the likelihood of a swift miner reception. While we find that 88.80% of the successful arbitrage transactions are accompanied by BRF, we cannot provide quantitative insights to what degree BRF improves the success-rate of back-running.

To quantify the network layer overhead, we identify past arbitrages on-chain and correlate the dropped transactions on the P2P network provided by our network listening node. We find that one of the most amplified flooding events entails 358 transactions on the network layer for a single arbitrage opportunity. These back-running transactions are identical, except the last byte of the transaction message, floods 65.7kb of data traversing the P2P network. This is equivalent to $1.5\times$ the average block size on the 1st of March 2021[11]. Only one of these transactions is confirmed on-chain[12], classified as a failed arbitrage attempt by us. The remaining 357 transactions have a conflicting nonce with the confirmed transaction, and therefore discarded. We observe that back-run flooding is comparatively cheap because bots issue conflicting MEV transactions (e.g., with the same nonce), while only one transaction is mined.

VI. Arbitrage/Routing Among N AMMs

In this section, we shed light on the performance of A^{2}MM when peered with N AMM markets (abbreviated as N-A^{2}MM). While Section B in the appendix provides an optimal arbitrage strategy of 2-A^{2}MM, Algorithm 1 presents our sub-optimal two-point arbitrage strategy for N-A^{2}MM, where $N > 2$. Intuitively, our strategy starts with the two AMMs offering the best and worst prices, and gradually narrows the price gap through arbitrage. Along this narrowing process, if the prices of a group of AMMs are synchronized, we aggregate their liquidity and treat them virtually as a single exchange. Executing a swap on a virtually aggregated exchange is equivalent to performing routing, where the trade volume is routed to each of the underlying AMM based on their liquidity. Our strategy, therefore, translate the arbitrage problem of N-A^{2}MM into 2-A^{2}MM sub-problems.

To ease the reader's understanding, we visualize the arbitrage process among three AMMs in Figure 12. The three AMMs we consider have prices sorted in ascending order (p_1, p_2, and p_3 respectively). Our algorithm hence performs arbitrage by considering only AMM 1 and 3 first. As the price gap narrows, we can encounter three different cases. In the first case, the price of AMM 1 increases from p_1 to p_2, which is synchronized with the price of AMM 2. Our algorithm then aggregates these two exchanges, and continues the arbitrage process between the newly aggregated virtual AMM and AMM 3. The second case is the symmetric to the first case, where the price of AMM 3 falls from p_3 to p_2, and our algorithm aggregates AMM 2 and 3. In the last case, (due to fees) the prices of all three AMMs are not synchronized, therefore we do not aggregate any AMMs.

Table VII shows the cost of Algorithm 1 among N constant product AMMs. We estimate that the transaction cost of 3-A^{2}MM is $1.7\times$ the cost for 2-A^{2}MM. We estimate the costs by applying linear interpolation based on our empirical cost evaluation from Table I.

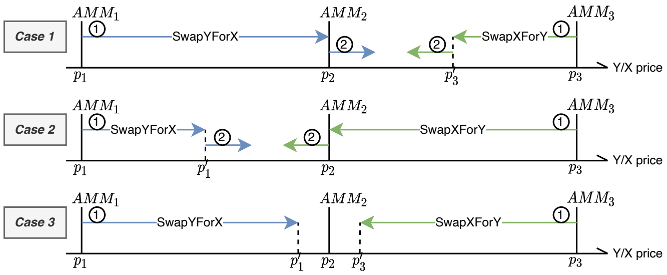

Fig. 12: Visualization of the two-point arbitrage process among three AMMs, which performs at most 3 swaps (either Swap$Y$to$X$ or Swap$X$to$Y$). In the first case, the liquidity from AMM_1 and AMM_2 are aggregated for Swap$Y$to$X$. The second case is symmetric to the first. In the third case, the arbitrage does not trigger AMM aggregation.

VII. Related Work

AMM: The literature proposes various blockchain-based exchange models covering limit order book models [5], [42], [39],

[11] https://etherscan.io/chart/blocksize

[12] 0x49bc22c9c45d31064f3cf7f7bd5e1494439603d4f6e809b0a715bc08d1b585c8

	Number of synchronized AMMs		
Type	2	3	$N > 3$
Number of			
Arbitrage computation	1	2	$N-1$
Synchronize volume	0	1	$2N-5$
Swaps	2	3	N
Cost over AMM average swap cost			
Routing to one AMM	$\underline{17.80\%}$	$\underline{17.80\%}$	$\underline{17.80\%}$
Arbitrage computation	$\underline{42.42\%}$	84.84%	$42.42\% \times (N-1)$
Threshold computation	N/A	$\underline{17.80\%}$	$17.80\% \times (2N-5)$
Swap execution	100%	150%	200%
Total cost	160.22%	270.44%	217.80% $+42.42\% \times (N-1)$ $+17.80\% \times (2N-5)$

TABLE VII: Cost prediction of performing two-point arbitrages among multiple AMMs when a user sells asset X to purchase asset Y. The underlined cost ratios are taken from our two-point arbitrage evaluation (cf. Table I). The synchronize volume quantifies the amount of trading volume required to synchronize the asset price among multiple AMMs. We estimate that the cost of synchronize volume is similar to the cost of a single optimal routing.

auctions [24], payment channel [41] and trusted hardware [16] were proposed in the literature. Uniswap is to date the most actively used constant product AMM, while alternative weighted AMMs emerged [1].

Arbitrage: Identifying arbitrage opportunities is extensively studied in traditional, centralized finance (or CeFi) [19], [21]. One common methodology is to create a graph of all pairwise assets that can be traded to use a greedy search strategy, such as the Bellman-Ford-Moore algorithm, to search the trading space. For instance, the Bellman-Ford-Moore algorithm operates with a complexity of $O(E*V)$ in a graph of E edges and V vertices. Such a greedy search methodology aims to create a circular, profitable trading opportunity. Greedy search approaches are restricted to actions such as trade asset X for Y. However, because a greedy search algorithm only follows the locally optimal choice at each action, it might fail to explore and find profitable trading strategies. Zhou et al. show two mechanisms to automatically discover profitable arbitrage opportunities in the intertwined DeFi contract graph [52]. Bartoletti et al. distill fundamental structural and economic aspects of AMMs, and in particular discuss the arbitrage problem [14].

Front-Running and Miner Extractable Value: Bonneau *et al.* [17], introduce the concept of bribery attack, which incentivizes miners to adopt a blockchain fork instead of the longest chain. Daian *et al.* [22], introduce the concept of gas price auctions (PGA) among trading bots as well as the concept of MEV. MEV widens the variance of block rewards, encourages both bribery and under-cutting attacks [17], [20]. The literature captures front-running by allowing a “rushing adversary” to interact with a protocol [15]. Previous studies [13], [53] suggest that HFT performance is strongly associated with latency and execution speed. The (financial) high-frequency trading (HFT) literature [7], [44] explores several trading strategies and their economic impact, such as arbitrage, news reaction strategies, etc. in traditional markets. Most of the traditional finance market strategies are applicable to AMM and decentralized exchanges [22], [8], [47], [46], [53], [52].

Eclipse Attacks: Strategically placed blockchain network nodes may control when and if miners receive transactions, affecting the transaction execution time [43], [34], [31], [33].

Malpractices on Exchanges: Malpractices on financial exchanges is a well-studied domain. Besides the traditional market manipulation techniques [36] (such as cornering, front-running, and pump-and-dump schemes), previous works [40] studies techniques such as spoofing, pinging, and mass misinformation, which leverage, e.g., social media, artificial intelligence, and natural language processing. Techniques were shown to deceive HFT algorithms [9]. To counterbalance this inherent trust, regulators conduct periodic and costly manual audits of banks, brokers, and exchanges to unveil potential misbehavior. Because DEXes operate under weak identities and censorship resilience (from both the creators, users, and miners), regulators may face challenges to impose anti-money laundering legislation.

VIII. Discussion

We hope that our work engenders a wider corpus of orthogonal blockchain application designs which take into account the nature of the underlying ledger. We would like to emphasize that our work is based on a non-optimized prototype implementation which can likely be improved through additional engineering efforts.

Our evaluation shows that A^{2}MM does lower the required exchange block-space by 32.8%. As such, A^{2}MM classifies as a scaling solution for both the network as well as the blockchain layer. While most existing backward-compatible scaling solutions such as payment channels, off-chain hubs, etc [32] provide weaker security guarantees, A^{2}MM inherits as a decentralized application the native blockchain security properties, and moreover improves the security of the blockchain consensus as shown in this paper.

IX. Conclusion

By means of the realization that one blockchain should only operate at most one AMM exchange, we design a novel A^{2}MM exchange, which allows exchange users to atomically extract Miner Extractable Value, instead of leaving such opportunity to others. We show how A^{2}MM can avoid two-point arbitrage MEV overhead on the P2P network and the blockchain transaction space. Reducing such overhead allows to strengthen the blockchain’s consensus security, without resorting to centralized relayer which undermine the very reason permissionless blockchains exist.

While A^{2}MM inherently takes advantage of the atomic nature of blockchain transactions for arbitrage and routing, our proposal can serve as inspiration to design further MEV-friendly DeFi protocols, e.g., for liquidations in lending markets. We hope that our work provides insights into a previously unconsidered and orthogonal design space for secure DeFi protocols which sustainably recognize the decentralized characteristics of permissionless ledgers.

References

[1] Balancer Exchange. https://balancer.finance/whitepaper/. accessed 21 March, 2021.

[2] Sushiswap. https://sushi.com/. accessed 21 March, 2021.

[3] Uniswap. https://uniswap.org/. accessed 21 March, 2021.

[4] Foreign exchange manipulation: FINMA issues six industry bans, 2019.

[5] Kyber: An on-chain liquidity protocol. Technical report, Kyber Network, April 2019.

[6] 1inch - what are private transactions and how they work?, 2020. https://help.1inch.exchange/en/articles/4695716-what-are-private-transactions-and-how-they-work.

[7] James J Angel and Douglas McCabe. Fairness in financial markets: The case of high frequency trading. *Journal of Business Ethics*, 112(4):585–595, 2013.

[8] Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of uniswap markets. *arXiv preprint arXiv:1911.03380*, 2019.

[9] Jakob Arnoldi. Computer algorithms, market manipulation and the institutionalization of high frequency trading. *Theory, Culture & Society*, 33(1):29–52, 2016.

[10] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International conference on principles of security and trust*, pages 164–186. Springer, 2017.

[11] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.

[12] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, 2019.

[13] Matthew Baron, Jonathan Brogaard, Björn Hagströmer, and Andrei Kirilenko. Risk and return in high-frequency trading. *Journal of Financial and Quantitative Analysis*, 54(3):993–1024, 2019.

[14] Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of automated market makers in defi. *arXiv preprint arXiv:2102.11350*, 2021.

[15] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 307–323. Springer, 1992.

[16] Iddo Bentov, Yan Ji, Fan Zhang, Lorenz Breidenbach, Philip Daian, and Ari Juels. Tesseract: Real-time cryptocurrency exchange using trusted hardware. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1521–1538, 2019.

[17] Joseph Bonneau. Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.

[18] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 104–121. IEEE, 2015.

[19] Mao-cheng Cai and Xiaotie Deng. Approximation and computation of arbitrage in frictional foreign exchange market. *Electronic Notes in Theoretical Computer Science*, 78:293–302, 2003.

[20] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016.

[21] Zhenyu Cui and Stephen Taylor. Arbitrage detection using max plus product iteration on foreign exchange rate graphs. *Finance Research Letters*, 35:101279, 2020.

[22] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges. *IEEE Symposium on Security and Privacy (SP)*, 2020.

[23] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Conference on Peer-to-Peer Computing*, pages 1–10, 2013.

[24] DutchX, July 2019. accessed 12 November, 2019, https://github.com/gnosis/dx-docs.

[25] Michael Egorov. Stableswap-efficient mechanism for stablecoin liquidity. *URl: https://medium. com/yield-protocol/introducing-ydai-43a727b96fc7*, 2019.

[26] Oğuzhan Ersoy, Zhijie Ren, Zekeriya Erkin, and Reginald L Lagendijk. Transaction propagation on permissionless blockchains: incentive and routing mechanisms. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 20–30. IEEE, 2018.

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

[28] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[29] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. Decentralization in bitcoin and ethereum networks. *arXiv preprint arXiv:1801.03998*, 2018.

[30] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 3–16. ACM, 2016.

[31] Arthur Gervais, Hubert Ritzdorf, Ghassan O Karame, and Srdjan Capkun. Tampering with the delivery of blocks and transactions in bitcoin. In *Conference on Computer and Communications Security*, pages 692–705. ACM, 2015.

[32] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Off The Chain Transactions. *IACR Cryptology ePrint Archive*, 2019:360, 2019.

[33] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015.

[34] Sebastian Henningsen, Daniel Teunis, Martin Florian, and Björn Scheuermann. Eclipsing ethereum peers with false friends. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 300–309. IEEE, 2019.

[35] Eyal Hertzog, Guy Benartzi, and Galia Benartzi. Bancor protocol. 2017.

[36] Robert A Jarrow. Market manipulation, bubbles, corners, and short squeezes. *Journal of financial and Quantitative Analysis*, 27(3):311–336, 1992.

[37] Lucianna Kiffer, Asad Salman, Dave Levin, Alan Mislove, and Cristina Nita-Rotaru. Under the hood of the ethereum gossip protocol.

[38] Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. Measuring Ethereum network peers. In *Proceedings of the Internet Measurement Conference 2018*, pages 91–104. ACM, 2018.

[39] Aurora Labs. Idex: A real-time and high-throughput ethereum smart contract exchange. Technical report, January 2019.

[40] Tom CW Lin. The new market manipulation. *Emory LJ*, 66:1253, 2016.

[41] Xuan Luo, Wei Cai, Zehua Wang, Xiuhua Li, and CM Victor Leung. A payment channel based hybrid decentralized ethereum token exchange. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 48–49. IEEE, 2019.

[42] MakerDao. Intro to the oasisdex protocol, September 2019. accessed 12 November, 2019, https://github.com/makerdao/developerguides/blob/master/Oasis/intro-to-oasis/intro-to-oasis-maker-otc.md.

[43] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. *IACR Cryptology ePrint Archive*, 2018(236), 2018.

[44] Albert J Menkveld. The economics of high-frequency trading: Taking stock. *Annual Review of Financial Economics*, 8:1–24, 2016.

[45] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[46] Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? *arXiv preprint arXiv:2101.05511*, 2021.

[47] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. *arXiv preprint arXiv:2003.03810*, 2020.

[48] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of financial studies*, 34(3):1156–1190, 2021.

[49] Fabian Schär. Decentralized finance: On blockchain-and smart contract-based financial markets. *Available at SSRN 3571335*, 2020.

[50] Karl Wüst and Arthur Gervais. Do you need a Blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE, 2018.

[51] Jiahua Xu, Nazariy Vavryk, Krzysztof Paruch, and Simon Cousaert. Sok: Automated market maker (amm) based decentralized exchanges (dexs). *arXiv preprint arXiv:2103.12732*, 2021.

[52] Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. On the just-in-time discovery of profit-generating transactions in defi protocols. *IEEE Symposium on Security and Privacy*, 2021.

[53] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. *IEEE Symposium on Security and Privacy*, 2021.

Appendix A
MEV Overhead Taxonomy

In the following, we provide a high-level taxonomy of the different sources of technical overhead introduced by MEV opportunities. We primarily differentiate between MEV opportunities on *(i)* the not-yet-confirmed network layer state and on *(ii)* the confirmed blockchain state. Note that we ignore the existence of blockchain forks for simplicity. We consider the MEV aware bots to not being miners, while the following reasoning also applies to miners extracting MEV.

Confirmed Block State MEV: An MEV extraction bot can choose to only act on a confirmed blockchain state, i.e., once a block is mined. Once a block at height i is received by the bot, the bot attempts to front-run all other transactions in the next block $i+1$. Confirmed state front-running is destructive, meaning that the bot bears no consideration to the subsequent transactions in the same block [46], [27].

Unconfirmed Block State MEV: An MEV extraction bot may observe the unconfirmed blockchain transactions and anticipate how the next miner would order these transactions within a block (e.g., given the paid transaction fees). Based on the anticipated transaction ordering, the arbitrageur then verifies whether new MEV opportunities surfaced. If an MEV opportunity is found, the bot ideally issues a back-running arbitrage transaction as an exploit [46], [27].

Priority Gas Auction (PGAs) Overhead: PGA is the process by which MEV aware bots are competitively bidding on transaction fees to obtain a specific transaction position (usually the first) in the next block. Because a transaction in the miner's mempool can be replaced before it is confirmed, PGA bots usually emit a new transaction with higher bids to replace their previous transaction [22], [53]. Although the replaced transaction is dropped by the network eventually after the new transaction is confirmed, the replaced transaction is still broadcasted on the network layer. Therefore PGA causes an overhead on the blockchain network layer.

Block-space Overhead: Trading bots increasingly extract MEV with optimal parameters [47], bequeathing no revenue for following MEV bot transactions, which should then either revert with an error or fail silently. We classify failed successful MEV transactions as on-chain MEV overhead.

Appendix B
Implementation

In this section, we present a concrete A^{2}MM implementation with two constant product AMM DEXes, namely Uniswap V2 and Sushiswap. Both these two exchanges follow a constant product formula, with a commission fee of 0.3% (cf. Equation 16).

$$\begin{aligned} p_{X \to Y}(x, y, \delta_x) &= y - \frac{x \cdot y}{x + \delta_x \cdot (1 - 0.3\%)} \\ p_{Y \to X}(x, y, \delta_y) &= x - \frac{x \cdot y}{y + \delta_y \cdot (1 - 0.3\%)} \end{aligned} \tag{16}$$

In the following, we denote the Uniswap and Sushiswap (X, Y) market as DEX 1 and 2, where the price of AMM 1 is greater than or equal to the price of AMM 2 for Swap$X$to$Y$ (i.e., $\frac{y_1}{x_1} \geq \frac{y_2}{x_2}$). We use (x_1, y_1) and (x_2, y_2) to denote the states of DEX 1 and DEX 2 respectively.

A. Route$X$to$Y$

As we have shown in Section III-E, the optimal routing strategy is to greedily route the trading volume to AMM 1 until the prices of both markets are synchronized. After the price synchronization, the remaining volume is routed to both AMM 1 and 2, while keeping the prices the same (cf. Theorem 1).

In Equation 17, we compute the threshold ($\bar{\delta_x}$), such that the prices between AMMs 1 and 2 will be synchronized after swapping exactly $\bar{\delta_x}$ of asset X for asset Y on AMM 1.

$$\begin{aligned} &\frac{y_1'}{x_1'} = \frac{y_2}{x_2} \Longleftrightarrow \frac{\frac{x_1 \cdot y_1}{x_1 + \bar{\delta_x} \cdot 0.997}}{x_1' + \bar{\delta_x}} = \frac{y_2}{x_2} \\ &\Longleftrightarrow \bar{\delta_x} \approx \frac{1.002(\sqrt{x_1 y_2 \left(2.257 \cdot 10^{-6} x_1 y_2 + x_2 y_1\right)} - x_1 y_2)}{y_2} \end{aligned} \tag{17}$$

We now consider the optimal routing strategy if the prices between AMMs 1 and 2 are synchronized (i.e., $\frac{y_1}{x_1} = \frac{y_2}{x_2}$). We use $q = \frac{x_1}{x_2} = \frac{y_1}{y_2}$ to denote the ratio of funds between the two DEXes. In Equation 18, we compute the optimal routing ratio k given that the liquidity taker trades δ_x amount of asset X, where we route $k \cdot \delta_x, (1-k) \cdot \delta_x$ to AMMs 1 and 2 respectively.

$$\begin{aligned} \frac{y_1'}{x_1'} = \frac{y_2'}{x_2'} &\Longleftrightarrow \frac{\frac{x_1 \cdot y_1}{x_1 + k \cdot \delta_x \cdot 0.997}}{x_1 + k \cdot \delta_x} = \frac{\frac{x_2 \cdot y_2}{x_2 + (1-k) \cdot \delta_x \cdot 0.997}}{x_2 + (1-k) \cdot \delta_x} \\ &\Longleftrightarrow \frac{\frac{q^2 \cdot x_2 \cdot y_2}{q \cdot x_2 + k \cdot \delta_x \cdot 0.997}}{q \cdot x_2 + k \cdot \delta_x} = \frac{\frac{x_2 \cdot y_2}{x_2 + (1-k) \cdot \delta_x \cdot 0.997}}{x_2 + (1-k) \cdot \delta_x} \\ &\Longleftrightarrow \frac{x_2 + (1-k) \cdot \delta_x \cdot 0.997}{x_2 + \frac{k}{q} \cdot \delta_x \cdot 0.997} = \frac{x_2 + \frac{k}{q} \cdot \delta_x}{x_2 + (1-k) \cdot \delta_x} \\ &\Longleftarrow 1 - k = \frac{k}{q} \Longleftrightarrow k = \frac{q}{1+q} \end{aligned} \tag{18}$$

Therefore, the optimal routing strategy routes $\bar{\delta_x}$ to AMM 1 first, such that the prices between the two exchanges are synchronized. The routing strategy then routes $\frac{q}{1+q}$ of the remaining liquidity to AMM 1, and $\frac{1}{1+q}$ to AMM 2 (cf. Equation 18).

B. ArbitrageFor$X$

In the following, we derive profitable arbitrage constraints among two constant product AMM exchanges (e.g., Uniswap and Sushiswap). The constraints are mathematically simple, such that a smart contract derives it at low costs on-chain. We also derive the formulas to calculate optimal two-point arbitrage parameters to maximize arbitrage revenue. Equation 19 shows the specific arbitrage objective function for two constant product AMMs, derived by substituting Equation 16 into Equation 9.

$$\text{maximize} \quad \delta'_x - \delta_x = p^{Y\to X}(x_2, y_2, p^{X\to Y}(x_1, y_1, \delta_x)) - \delta_x \quad (19)$$

To find the optimal arbitrage parameter ($\delta_x^{optimal}$), we solve the derivative of the objective function $\frac{d}{dx}(\delta'_x - \delta_x) = 0$. Equation 20 shows the only positive solution for $\delta_x^{optimal}$.

$$\delta_x^{optimal} \approx \frac{1.003\left(-1000x_1y_2 + 997\sqrt{x_1x_2y_1y_2}\right)}{997y_1 + 1000y_2} \quad (20)$$

In Equation 21, we substitute $\delta_x^{optimal}$ into the objective function to derive the optimal revenue.

$$\begin{aligned}\delta'_x - \delta_x \approx & -\frac{x_2y_2}{-\frac{0.997x_1y_1}{x_1 - \frac{0.997(c_1x_1y_2 - c_2\sqrt{x_1x_2y_1y_2})}{997y_1+1000y_2}} + 0.997y_1 + y_2} \\ & + x_2 + \frac{c_1x_1y_2 - c_2\sqrt{x_1x_2y_1y_2}}{997y_1 + 1000y_2} \\ \text{where} \quad & c_1 \approx 1003.009 \\ & c_2 \approx 1000.000\end{aligned} \quad (21)$$

In Equation 22, we find the constraint for the arbitrage opportunity to be profitable, without considering transaction fees.

$$\delta'_x - \delta_x > 0 \implies \frac{y_2}{x_2} < 0.994\frac{y_1}{x_1} \quad (22)$$

We, therefore, apply Equation 22 to verify whether arbitrage is profitable given a blockchain state and use the optimal parameters (cf. Equation 20) to extract the maximum revenue.

Appendix C
Miner Hashing Power

Table VIII shows the hashing power we extract from various sources to simulate the P2P network for Ethereum[13], Bitcoin[14], Litecoin[15] and Dogecoin[16].

Rank	Ethereum	Bitcoin	Litecoin	Dogecoin
1	24.3%	17.9%	16.0%	14.9%
2	19.3%	15.5%	14.4%	13.61%
3	10.4%	11.9%	14.0%	13.38%
4	5.8%	11.4%	12.2%	12.58%
5	4.6%	9.9%	11.4%	11.46%
6	4.3%	8.7%	10.2%	10.74%
7	3.8%	8.1%	9.2%	8.68%
8	2.8%	4.3%	7.4%	7.35%
9	2.6%	2.7%	1.8%	1.47%
10	2.5%	2.5%	1.2%	0.73%

TABLE VIII: The hashing power distribution for Ethereum, Bitcoin, Litecoin and Dogecoin as of April 2021.

Pct %	0%	10%	33%	50%	67%	90%	100%
[38]	-	99	151	208	231	285	-
[29]	-	92	125	152	200	276	-
This work	0	95.5	138	180	215.5	280/5	300

TABLE IX: We base the latency distribution(ms) in this work on the mean statistics of the Ethereum P2P network provided by related works [38], [29].

Appendix D
Sub-optimal two-points arbitrage for N-A^{2}MM

Algorithm 1 shows the sub-optimal ArbitrageFor$Y$ strategy among $N+1$ AMMs on X/Y market.

Algorithm 1: Sub-optimal ArbitrageFor$Y$ strategy among $N+1$ AMMs on X/Y market. Our strategy iteratively performs three steps to extract arbitrage revenue, namely *(i)* AMMs aggregation; *(ii)* arbitrage computation; *(iii)* threshold computation; *(iv)* swap execution, until the price of all AMMs are leveled.

```
17 $M[0\dots N] \leftarrow$ AMMs with ascending $Y/X$ price;
$l \leftarrow 0$ ; $r \leftarrow N$ ;
while True do
    // Aggregation
    $M_L \leftarrow$ aggregate exchanges $M[0]$ to $M[l]$;
    $M_R \leftarrow$ aggregate exchanges $M[r]$ to $M[N]$;
    if arbitrage between$M_L$ and $M_R$ is profitable? then
        // Arbitrage computation
        Simulate arbitrage between $M_L$ and $M_R$;
        $p_{M_L} \leftarrow$ price of $M_L$ after arbitrage simulation;
        $p_{M_R} \leftarrow$ price of $M_R$ after arbitrage simulation;
        $b_L^{\text{shift}} \leftarrow (p_{M_L} > p_{M[l+1]}) \wedge ((l+1) < r)$;
        $b_R^{\text{shift}} \leftarrow (p_{M_R} > p_{M[r-1]}) \wedge (l < (r-1))$;
        // Synchronize volume
        if $b_L^{shift}$ then
            $\delta_x^l \leftarrow$ such that $p_{M_L} == p_{M[l+1]}$ if
             ArbitrageFor$Y(\delta_x^l, M[l], M[r])$ is executed;
        if $b_R^{shift}$ then
            $\delta_x^r \leftarrow$ such that $p_{M_R} == p_{M[r-1]}$ if
             ArbitrageFor$Y(\delta_x^r, M[l], M[r])$ is executed;
        // Swap execution
        if $b_L^{shift}$ and $\delta_x^l \leq \delta_x^r$ then
            ArbitrageFor$Y(\delta_x^l, M[l], M[r])$;
            continue;
        if $b_R^{shift}$ and $\delta_x^l \geq \delta_x^r$ then
            ArbitrageFor$Y(\delta_x^r, M[l], M[r])$;
            continue;
    else
        break;
```

[13] https://etherscan.io/stat/miner?blocktype=blocks, accessed Apr 2021

[14] https://btc.com/stats/pool, accessed Apr 2021

[15] https://www.litecoinpool.org/pools, accessed Apr 2021

[16] https://explorer.viawallet.com/doge/pool, accessed Apr 2021

Appendix E
Past Arbitrage Volume Opportunities

In the following, we quantify the volume and number of transactions performing two-point arbitrage on past blockchain data from block $10,794,261$ (4th September, 2020, Sushiswap's deployment) to block $12,000,000$ (8th March, 2021) (186 days).

Arbitrage Heuristics: We adjust the heuristics proposed by Qin *et al.* [46] to detect past extracted two-point arbitrages. Recall that every $\mathsf{ArbitrageFor}X$ consists of two state transitions (cf. Equation 2). In the following we denote these two transitions as $\mathsf{Swap}X\mathsf{to}Y(\delta_x^1, \delta_y^1)$ and $\mathsf{Swap}Y\mathsf{to}X(\delta_y^2, \delta_x^2)$.

Heuristic 1 We assume that the arbitrageurs attempt to minimize their risks, and therefore execute both $\mathsf{Swap}X\mathsf{to}Y$ and $\mathsf{Swap}Y\mathsf{to}X$ atomically in the same transaction. Note that, unlike previous work [46], we do not constrain the execution order of $\mathsf{Swap}X\mathsf{to}Y$ and $\mathsf{Swap}Y\mathsf{to}X$.

Heuristic 2 The output (δ_y^1) of $\mathsf{Swap}X\mathsf{to}Y$ must be greater than the input (δ_y^2) of $\mathsf{Swap}Y\mathsf{to}X$.

Heuristic 3 The input (δ_x^1) of $\mathsf{Swap}X\mathsf{to}Y$ must be less than the output (δ_x^2) of $\mathsf{Swap}Y\mathsf{to}X$. This implicitly assumes that an arbitrageur asserts a positive revenue in the smart contract, and reverts otherwise (ignoring the transaction fees).

Empirical Results: We consider all assets and markets on Uni- and Sushiswap over 186 days, from block $10,794,261$ (4th September, 2020, Sushiswap's deployment) to block $12,000,000$ (8th March, 2021). Among the $22,232,144$ Uni/Sushiswap related transactions, we identify a total of $164,345$ (0.7%) successful two-point arbitrage trades. These arbitrage activities realize a revenue of $28,956$ ETH ($25,541,382$ DAI), contributing $1.8B$ USD of trading volume to each of the two exchanges (1.74% and 5% of Uni- and Sushiswap's total transaction volume, respectively).

To better understand these arbitrage revenues, we deduct the transaction fees (gas costs) from these transactions. We find that $131,896$ (80.07%) of the arbitrages are profitable, paying a total, $4,348.11$ ETH (32 billion gas) in transaction fees. On average, two-point arbitrage yields an average revenue of $118,626$ and $25,730$ USD per day for arbitrageurs and miners, respectively.

	Trading Volume		**Arbitrage Volume (%)**	
Month	Uniswap	Sushiswap	Uniswap	Sushiswap
20-09	12.2B	2.2B	215.4M(1.77%)	215.3M(9.73%)
20-10	9.1B	855.0M	89.4M(0.99%)	89.2M(10.44%)
20-11	9.6B	2.0B	117.8M(1.23%)	117.3M(5.92%)
20-12	11.7B	3.0B	153.9M(1.31%)	151.6M(5.01%)
20-01	25.3B	11.7B	452.7M(1.79%)	442.2M(3.77%)
20-02	32.5B	14.2B	730.9M(2.25%)	724.1M(5.11%)

TABLE X: Trading volume, two-point arbitrage on Uniswap, Sushiswap. For example, in October 2020 arbitrage activities amount to a total volume of $855.0M$ USD, 10.44% of the entire trading volume on Sushiswap ($14.2B$ USD).

Heuristic Limitations: Heuristic 1 assumes that arbitrageurs perform arbitrage only within atomic transactions, as in to minimize execution risks. Naturally, some arbitrageurs may not perform atomic arbitrage, especially when not colluding with miners. Heuristic 1 can therefore introduce false negatives by not capturing seemingly riskier arbitrage. Heuristics 2 and 3 will only detect arbitrage where asset X increases but asset Y does not decrease. While we lack quantitative insights on the heuristic accuracy, we choose to introduce relatively strict heuristics to reduce our false positive rate at the cost of underestimating the overall arbitrage transactions.