Diamonds are Forever, Loss-Versus-Rebalancing is Not

Conor McMenamin[1,2], Vanesa Daza[1,3], and Bruno Mazorra[1,2]

[1] Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain
[2] NOKIA Bell Labs, Nozay, France
[3] CYBERCAT - Center for Cybersecurity Research of Catalonia

email: (first name).(last name)@upf.edu

Abstract. The always-available liquidity of automated market makers (AMMs) has been one of the most important catalysts in early cryptocurrency adoption. However, it has become increasingly evident that AMMs in their current form are not viable investment options for passive liquidity providers. This is because of the cost incurred by AMMs providing stale prices to arbitrageurs against external market prices, formalized as loss-versus-rebalancing (LVR) [Milionis et al., 2022].
In this paper, we present Diamond, an automated market making protocol that aligns the incentives of liquidity providers and block producers in the protocol-level retention of LVR. In Diamond, block producers effectively auction the right to capture any arbitrage that exists between the external market price of a Diamond pool, and the price of the pool itself. The proceeds of these auctions are shared by the Diamond pool and block producer in a way that is proven to remain incentive compatible for the block producer. Given the participation of competing arbitrageurs, LVR is effectively prevented in Diamond. We formally prove this result, and detail an implementation of Diamond. We also provide comparative simulations of Diamond to relevant benchmarks, further evidencing the LVR-protection capabilities of Diamond. With this new protection, passive liquidity provision on blockchains becomes rationally viable, beckoning a new age for decentralized finance.

Keywords: Extractable Value · Decentralized Exchange · Incentives · Blockchain

1 Introduction

Constant function market makers (CFMMs) such as Uniswap [22] have emerged as the dominate class of automated market making (AMM) protocols. CFMMs

This Technical Report is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 814284

offer several key advantages for decentralized liquidity provision. They are efficient computationally, have minimal storage needs, matching computations can be done quickly, and liquidity providers can be passive. Thus, CFMMs are uniquely suited to the severely computation- and storage-constrained environment of blockchains.

Unfortunately, the benefits of CFMMs are not without significant costs. These costs are definitively formalized and quantified in [17] as *loss-versus-rebalancing* (LVR). It is proved that as the underlying price of a swap moves around in real-time, the discrete-time progression of AMMs leave arbitrage opportunities against the AMM. In centralized finance, market makers typically adjust to new price information before trading. This comes at a considerable cost to AMMs (for CFMMs, [17] derives the cost to be quadratic in realized moves), with similar costs for AMMs derived quantitatively in [18,7].

Unfortunately, these costs are being realized by liquidity providers in current AMM protocols. In Uniswap V3 [22], the most prominent AMM by transaction volume, the realized LVR from just May 2021 to September 2021 was quantified in [15] to be $\$61m$. Furthermore, volume trading on AMMs has been decreasing for multiple consecutive months (Figure 1), while toxic order flow (similar to LVR) is consistently profiting against AMM liquidity providers (Figure 2). All of these factors point towards unsatisfactory protocol design, and a dire need for an LVR-resistant automated market maker. In this paper, we provide Diamond, an AMM protocol which formally protects against LVR.

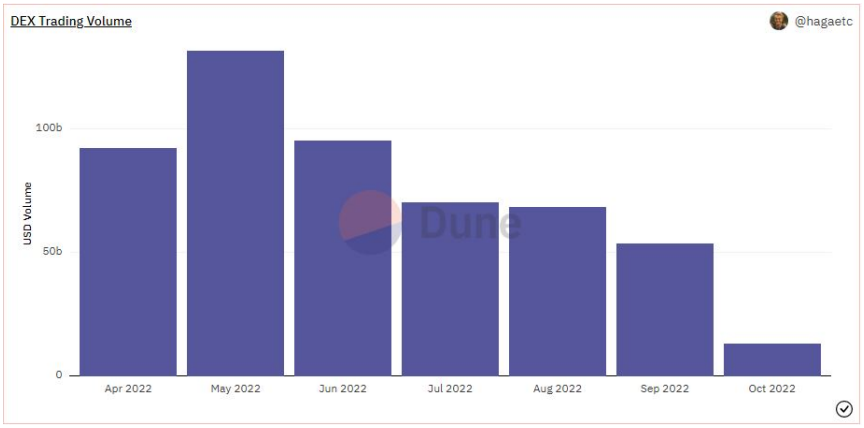

Fig. 1: DEX Trading Volumes [11].

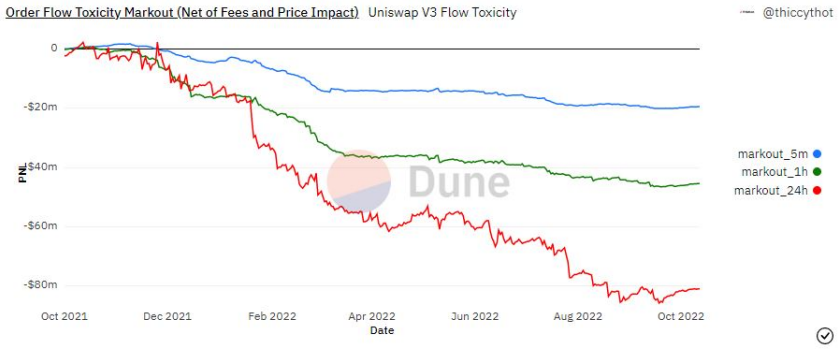

Fig. 2: Toxicity of Order Flow [21]. This graph aggregates the PnL of all trades on the Uniswap V3 WETH/USDC pool, measuring PnL of each order after 5 minutes, 1 hour, and 1 day. This shows that the liquidity providers are consistently losing money in the most prominent CFMM.

1.1 Our Contribution

We present Diamond, an AMM protocol which isolates the LVR being captured from a Diamond liquidity pool by arbitrageurs, and forces these arbitrageurs to repay some percentage of this LVR to the pool. As in typical CFMMs, Diamond pools are defined with respect to two tokens x and y. At any given time, the pool has reserves of R_x and R_y of both tokens, and some pool pricing function[4] $P(R_x, R_y)$. We demonstrate our results using the well-studied Uniswap V2 pricing function of $P(R_x, R_y) = \frac{R_x}{R_y}$.

We introduce the concept of its corresponding CFMM pool for each Diamond pool. For a Diamond pool with token reserves (R_x, R_y) and pricing function $P(R_x, R_y) = \frac{R_x}{R_y}$, the corresponding CFMM pool is the Uniswap V2 pool with reserves (R_x, R_y). If an arbitrageur tries to move the price of the corresponding CFMM pool adding Δ_x tokens and removing Δ_y, the same price is achieved in the Diamond pool by adding $(1-\alpha)\Delta_x$ tokens for some $\alpha > 0$, with α the *LVR rebate parameter*. The arbitrageur receives $(1-\alpha)\Delta_y$. In our framework, it must be that $P(R_x + (1-\alpha)\Delta_x, R_y - (1-\alpha)\Delta_y) < P(R_x + \Delta_x, R_y - \Delta_y)$, which also holds in our example, as $\frac{R_x+(1-\alpha)\Delta_x}{R_y-(1-\alpha)\Delta_y} < \frac{R_x+\Delta_x}{R_y-\Delta_y}$. A further θ_y tokens are removed from the Diamond pool to move the reserves to the same price as the corresponding CFMM pool, with these tokens added to the *vault*.

Half of the tokens in the vault are then periodically converted into the other token in one of the following ways:

1. An auction amongst arbitrageurs.
2. Converted every block by the arbitrageur at the final pool price. If the arbitrageur must buy $\frac{\theta}{2}$ tokens to convert the vault, the arbitrageur must simultaneously sell $\frac{\theta}{2}$ futures which replicate the price of the token to the pool. These futures are then settled periodically, either by

[4] See Equation 1 for a full description of pool pricing functions as used in this paper

(a) Auctioning the $\frac{\theta}{2}$ tokens corresponding to the futures to the arbitrageurs, with the protocol paying/collecting the difference.
(b) The use of a decentralized price oracle. In this paper, we consider the use of the settlement price of an on-chain frequent batch auction, such as [16], which is proven to settle at the external market price in expectancy.

Importantly, these auctions are not required for protocol liveness, and can be arbitrarily slow to settle. We prove that all of these conversion processes have 0 expectancy for the arbitrageur or Diamond pool, and prove that the LVR of a Diamond pool is $(1 - \alpha)$ of the corresponding CFMM pool. We describe an implementation of Diamond which isolates arbitrageurs from normal users. This ensures the protections of Diamond can be provided in practice while providing at least the same trading experience for normal users. Non-arbitrageur orders in a Diamond pool are performed identically to orders in the corresponding CFMM pool. We discuss practical considerations for implementing Diamond, including how decreasing the LVR rebate parameter during protocol inactivity ensures the protocol continues to process user transactions while still providing some level of LVR protection.

We present a series of experiments in Section 6 which isolate the benefits of Diamond. We compare a Diamond pool to its corresponding Uniswap V2 pool, as well as the strategy of holding the starting reserves of both tokens, demonstrating the power of Diamond. We isolate the effects of price volatility, LVR rebate parameter, pool fees, and pool duration on a Diamond pool. Our experiments provide statistically significant evidence that the relative value of a Diamond pool to its corresponding Uniswap V2 pool is increasing in each of these variables. These experiments further evidence the limitations of current CFMMs, and the potential of Diamond.

2 Related Work

There are many papers on the theory and design of AMMs, with some of the most important including [2,1,17,3,4]. The only peer-reviewed AMM design claiming protection against LVR [14] is based on live price oracles. The AMM must receive the price of a swap before users can interact with the pool. Such sub-block time price data requires centralized sources which are prone to manipulation, or require the active participation of AMM representatives, a contradiction of the passive nature of AMMs and their liquidity providers. We see this as an unsatisfactory dependency for DeFi protocols.

Attempts to provide LVR protection without explicit use of oracles either use predictive fees for all players [10] and/or reduce liquidity for all players through more complex constant functions [5]. Charging all users higher fees to compensate for arbitrageur profits reduces the utility of the protocol for genuine users, as does a generalized liquidity reduction. In Diamond, we only reduce liquidity for arbitrageurs (which can also be seen as an increased arbitrageur-specific fee), providing at least the same user experience for typical users as existing AMMs without LVR protection.

A recent proposed solution to LVR published in a blog-post [12] termed *MEV-capturing AMMs* (McAMMs) considers auctioning off the first transaction/series of transaction in an AMM among arbitrageurs, with auction revenue paid in some form to the protocol. Two important benefits of Diamond compared to the proposed McAMMs are the capturing of realized LVR in Diamond as opposed to predicted LVR in McAMMs, and decentralized access to Diamond compared to a single point of failure in McAMMs.

In McAMMs, bidders are required to predict upcoming movements in the AMM. Bidders with large orders to execute over the period (e.g. private price information, private order flow, etc.) have informational advantages over other bidders. Knowing the difference between expected LVR excluding this private information vs. true expected LVR allows the bidder to inflict more LVR on the AMM than is paid for. As this results in better execution for the winner's orders, this may result in more private order flow, which exacerbates this effect. Diamond extracts a constant percentage of the true LVR, regardless of private information. McAMMs also centralize (first) access control to the winning bidder. If this bidder fails to respond or is censored, user access to the protocol is prohibited/more expensive. Diamond is fully decentralized, incentive compatible and can be programmed to effectively remove LVR in expectancy. Future McAMM design improvements based on sub-block time auctions are upper-bounded by the current protection provided by Diamond.

3 Preliminaries

This section introduces the key terminology and definitions needed to understand LVR, the Diamond protocol, and the proceeding analysis. In this work we are concerned with a single swap between token x and token y. We use x and y subscripts when referring to quantities of the respective tokens. The external market price of a swap is denoted by a lowercase p, while pool prices/ price functions are denoted using an uppercase P, with the price of a swap quoted as the quantity of token x per token y.

3.1 Constant Function Market Makers

A CFMM is characterized by *reserves* $(R_x, R_y) \in \mathbb{R}^2_+$ which describes the total amount of each token in the pool. The price of the pool is given by *pool price function* $P : \mathbb{R}^2_+ \to \mathbb{R}$ taking as input pool reserves (R_x, R_y). P has the following properties:

$$\begin{aligned}
&\text{(a) } P \text{ is everywhere differentiable, with } \frac{\partial P}{\partial R_x} > 0,\ \frac{\partial P}{\partial R_y} < 0.\\
&\text{(b) } \lim_{R_x \to 0} P = 0,\ \lim_{R_x \to \infty} P = \infty,\ \lim_{R_y \to 0} P = \infty,\ \lim_{R_y \to \infty} P = 0.\\
&\text{(c) If } P(R_x, R_y) = p, \text{ then } P(R_x + c, R_y + cp) = p,\ \forall c > 0.
\end{aligned} \tag{1}$$

These are typical properties of price functions. Property (a) states the price of y is increasing in the number of x tokens in the pool and decreasing in the number of y tokens. Property (b) can be interpreted as any pool price value is reachable for a fixed R_x, by changing the reserves of R_y, and vice versa. Property (c) states that adding reserves to a pool in a ratio corresponding to the current price of the pool does not change the price of the pool. These properties trivially hold for the Uniswap V2 price function of $\frac{R_x}{R_y}$, and importantly allow us to generalize our results to a wider class of CFMMs.

For a CFMM, the *feasible set of reserves* C is described by:

$$C = \{(R_x, R_y) \in \mathbb{R}_+^2 : f(R_x, R_y) = k\} \tag{2}$$

where $f : \mathbb{R}_+^2 \to \mathbb{R}$ is the pool invariant and $k \in \mathbb{R}$ is a constant. The pool is defined by a smart contract which allows any player to move the pool reserves from the current reserves $(R_{x,0}, R_{y,0}) \in C$ to any other reserves $(R_{x,1}, R_{y,1}) \in C$ if and only if the player provides the difference $(R_{x,1} - R_{x,0}, R_{y,1} - R_{y,0})$.

Whenever an arbitrageur interacts with the pool, say at time t with reserves $(R_{x,t}, R_{y,t})$, we assume as in [17] that the arbitrageur maximizes their profits by exploiting the difference between $P(R_{x,t}, R_{y,t})$ and the external market price at time t, denoted p_t. To reason about this movement, we consider a *pool value function* $V : \mathbb{R}_+ \to \mathbb{R}$ defined by the optimization problem:

$$V(p_t) = \min_{(R_x, R_y) \in \mathbb{R}_+^2} p_t R_y + R_x, \text{ such that } f(R_x, R_y) = k \tag{3}$$

Given an arbitrageur interacts with the pool with external market price p_t, the arbitrageur moves the pool reserves to the (R_x, R_y) satisfying $V(p_t)$.

3.2 Loss-Versus-Rebalancing

LVR, and its prevention in AMMs is the primary focus of this paper. The formalization of LVR [17] has illuminated the true cost of providing liquidity in CFMMs. The authors of [17] provide various synonyms to conceptualize LVR. In this paper, we use the opportunity cost of arbitraging the pool against the external market price of the swap, which is proven to be equivalent to LVR in Corollary 1 of [17]. The LVR between two blocks B_t and B_{t+1} where the reserves of the AMM at the end of B_t are $(R_{x,t}, R_{y,t})$ and the external market price when creating block B_{t+1} is p_{t+1} is:

$$R_{x,t} + R_{y,t} p_{t+1} - V(p_{t+1}) = (R_{x,t} - R_{x,t+1}) + (R_{y,t} - R_{y,t+1}) p_{t+1}. \tag{4}$$

As this is the amount being lost to arbitrageurs by the AMM, this is the quantity that needs to be minimized in order to provide LVR protection. In Diamond, this minimization is achieved.

3.3 Auctions

To reason about the incentive compatibility of parts of our protocol, we outline some basic auction theory results.

First-price-sealed-bid-auction: There is a finite set of players $\mathcal{I}$ and a single object for sale. Each bidder $i \in \mathcal{I}$ assigns a value of X_i to the object. Each X_i is a random variable that is independent and identically distributed on some interval $[0, V_{max}]$. The bidders know its realization x_i of X_i. We will assume that bidders are risk neutral, that they seek to maximize their expected payoff. Per auction, each player submit a bid b_i to the auctioneer. The player with the highest bid gets the object and pays the amount bid. In case of tie, the winner of the auction is chosen randomly. Therefore, the utility of a player $i \in \mathcal{I}$ is

$$u_i(b_i, b_{-i}) = \begin{cases} \frac{x_i - b_i}{m}, & \text{if } b_i = \max_i\{b_i\}, \\ 0, & \text{otherwise} \end{cases}$$

where $m = |\text{argmax}_i\{b_i\}|$. In our protocol, we have an amount of tokens z that will be auctioned. This object can be exchanged by all players at the external market price p. In this scenario, we have the following lemma.

Lemma 1. *Let $\mathcal{I}$ be a set of players that can exchange at some market any amount of tokens x or y at the external market price p. If an amount z of either token are auctioned in a first-price auction, then the maximum bid of any Nash equilibrium is at least zp.*

Proof. By construction, we have that the support of X_i is lower bounded by zp. Therefore, in a second-price auction, in equilibrium, each player will bid at least, zp. Using the revenue equivalence theorem [13], we deduce that the revenue of the seller is at least zp obtaining the result.

In one variation of Diamond, it possible to use a periodically updated price oracle to ensure the incentive compatibility of the protocol. To instantiate a game-theoretically secure decentralized price oracles, we can use the settlement price of a decentralized frequent batch auction [16]. Frequent batch auctions have been proven to settle in expectancy at the external market price when the auction is run [6,16]. Such guarantees were originally intended to benefit the users of the frequent batch auction. However, given the settlement price of the auction has expectancy equal to the external market price at a specific time, we can use this as a practical and secure price oracle for settling derivatives depending on this price, without the need for centralized alternatives like Chainlink [8].

4 Diamond

This section introduces the Diamond protocol. When the core protocol of Section 4.2 is run, some amount of tokens are removed from the pool and placed in a *vault*. These vault tokens are eventually re-added to the pool through a conversion protocol. Sections 4.3 and 4.4 detail two conversion protocols which can be

run in conjunction with the core Diamond protocol. Which conversion protocol to use depends on the priorities of the protocol users, with a comparison of their trade-offs provided in Section 6. These trade-offs can be summarized as follows:

- The process of Section 4.3 ensures the available liquidity is strictly increasing in expectancy every block, and can be used in conjunction with a decentralized price oracle to ensure the only required participation of a arbitrageurs is in arbitraging the pool (see process 2 in Section 4.3).
- The process in Section 4.4 incurs less variance in the total value of tokens owned by the pool (see Figure 3), and involves a more straightforward use of an auction.

Section 4.5 formalizes the properties of Diamond, culminating in Theorem 1, which states that Diamond can be parameterized to reduce LVR arbitrarily close to 0. It is important to note that Diamond is not a CFMM, but the rules for adjusting pool reserves are dependent on a CFMM.

4.1 Model Assumptions

We outline here the assumptions used when reasoning about Diamond. Inkeeping with the seminal analysis of [17], we borrow a subset of the assumptions therein, providing here a somewhat more generalized model.

1. External market prices follow a martingale process.
2. The risk-free rate is 0.
3. There exists a population of arbitrageurs able to frictionlessly trade at the external market price, who continuously monitor and periodically interact with the CFMM pool.
4. No liquidity providers enter or leave the system.
5. An optimal solution (R_x^*, R_y^*) to Equation 3 exists for every external market price $p \geq 0$.

The use of futures contracts in one version of the Diamond protocol makes the risk-free rate an important consideration for implementations of Diamond. If the risk free rate is not 0, the PnL related to owning token futures vs. physical tokens must be considered. Analysis of a non-zero risk-free rate is beyond the scope of this paper.

4.2 Core Protocol

We now describe the core Diamond, which is run by all Diamond variations. A Diamond pool Φ is described by reserves (R_x, R_y), a pricing function $P()$, a pool invariant function $f()$, an *LVR-rebate parameter* $\alpha \in (0, 1)$, and *conversion frequency* $T \in \mathbb{N}$.

We define the *corresponding CFMM pool* of Φ, denoted *CFMM*(Φ), as the CFMM pool with reserves (R_x, R_y) whose feasible set is described by pool invariant function $f()$ and pool constant $k = f(R_x, R_y)$. Conversely, Φ is the *corresponding Diamond pool* of *CFMM*(Φ). It is important to note that *CFMM*(Φ)

changes every time the Φ pool reserves change. The protocol progresses in blocks, with one reserve update per block.

Consider pool reserves $(R_{x,0}, R_{y,0})$ in Φ and a player wishing to move the price of Φ to $P_1 \neq \frac{R_{x,0}}{R_{y,0}}$. For a player wishing to move the price of *CFMM*(Φ) to P_1 from starting reserves $(R_{x,0}, R_{y,0})$, let this require $\Delta_y > 0$ tokens to be added to *CFMM*(Φ), and $\Delta_x > 0$ tokens to be removed from *CFMM*(Φ). The same price in Φ is achieved by the following process[5]:

1. Adding $(1-\alpha)\Delta_y$ tokens to Φ and removing $(1-\alpha)\Delta_x$ tokens.
2. Removing $\delta_x > 0$ tokens such that:

$$P(R_{x,0} - (1-\alpha)\Delta_x - \delta_x, R_{y,0} + (1-\alpha)\Delta_y) = P_1.^{6} \tag{5}$$

These δ_x tokens are added to the *vault* of Φ.

After this process, let there be $(v_x, v_y) \in \mathbb{R}^2_+$ tokens in the vault of Φ. If $v_y p_1 > v_x$, add $(v_x, \frac{v_x}{p_1})$ tokens into Φ from the vault. Otherwise, add $(v_y p_1, v_y)$ tokens into Φ from the vault. This is a *vault rebalance*.

Every T blocks, after the vault rebalance, the protocol converts half of the tokens still in the vault of Φ (there can only be one token type in the vault after a vault rebalance) into the other token in Φ according to one of either conversion process 1 (Section 4.3) or 2 (Section 4.4). After the conversion process, all tokens still in the vault of Φ are added into the Φ pool.

4.3 Per-block Conversion vs. Future Contracts

After every arbitrage, the arbitrageur converts θ equal to half of the total tokens in the vault at the pool price P_c, equivalent to buying θ tokens for P_c. Simultaneously, the arbitrageur sells to the pool θ future contracts in the same token denomination at price P_c. Given the pool buys θ future contracts at conversion price P_c, and the futures settle at price p_T, the protocol wins $\theta(p_T - P_c)$.

These future contracts are settled every T blocks, with the net profit or loss being paid in both tokens, such that for a protocol settlement profit of PnL measured in token x and pool price P_T, the arbitrageur pays (s_x, s_y) with $PnL = s_x + s_y P_T$ and $s_x = s_y P_T$. These contracts can be settled in one of the following (non-exhaustive) ways to settle futures:

1. Every T blocks, an auction takes place to buy the offered tokens from the players who converted the pool at the prices at which the conversions took place. For a particular offer, a positive bid implies the converter lost/the pool won to the futures. In this case the converter gives the tokens to the

[5] If $\Delta_y > 0$ tokens are to be removed from *CFMM*(Φ) with $\Delta_x > 0$ tokens to be added in order to achieve P_1, then $(1-\alpha)\Delta_y$ tokens are removed from Φ and $(1-\alpha)\Delta_x$ tokens are added to Φ, with a further $\delta_y > 0$ removed from Φ and added to the vault such that $P(R_{x,0} + (1-\alpha)\Delta_x, R_{y,0} - (1-\alpha)\Delta_y - \delta_y) = P_1$.

[6] Achievable as a result of properties(a) and (b) of Equation 1.

auction winner, while the pool receives the winning auction bid. A negative bid implies the converter won/the pool lost to the futures. In this case, the converter must also give the tokens to the auction winner, while the pool must pay the absolute value of the winning bid to the auction winner.
2. Every T blocks, a blockchain-based frequent batch auction takes place in the swap corresponding to the pool swap. The settlement price of the frequent batch auction is used as the price at which to settle the futures.

4.4 Periodic Conversion Auction

Every T blocks, θ equal to half of the tokens in the vault are auctioned to all players in the system, with bids denominated in the other pool token. For winning bid b in token x (or token y), the resultant vault quantities described by $(s_x = b, s_y = \theta)$ (or $(s_x = \theta, s_y = b)$) are added to the pool reserves. In this case, unlike in Section 4.3, there are no restrictions placed on $\frac{s_x}{s_y}$. As such, they may be in a different ratio than the pool reserves.

4.5 Diamond Properties

This section outlines the key properties of Diamond. Omitted proofs are included in Appendix A due to space constraints. We first prove that both conversion process have at least 0 expectancy for the protocol.

Lemma 2. *Converting the vault every block vs. future contracts has expectancy of at least 0 for a Diamond pool.*

Lemma 3. *A periodic conversion auction has expectancy of at least 0 for a Diamond pool.*

Corollary 1. *Conversion has expectancy of at least 0 for a Diamond pool.*

With these results in hand, we now prove the main result of the paper. That is, the LVR of a Diamond pool is $(1 - \alpha)$ of the corresponding CFMM pool.

Theorem 1. *For a CFMM pool $CFMM(\Phi)$ with LVR of $L > 0$, the LVR of Φ, the corresponding pool in Diamond, has expectancy of at most $(1 - \alpha)L$.*

5 Implementation

We now detail an implementation of Diamond. In our implementation, we consider block producers as arbitrageurs, with names interchangeable, with block producers in Diamond not charged protocol fees. The main focus of our implementation is ensuring user experience in a Diamond pool is not degraded compared to the corresponding CFMM pool. To this point, applying an α-discount on every Diamond pool trade is not viable. To avoid this, we only consider LVR on a per-block, and not a per-transaction basis. Given the transaction sequence,

in/exclusion and priority auction capabilities of block producers, block producers can either capture the block LVR of a Diamond pool themselves, or effectively sell this right to other arbitrageurs.

From an implementation standpoint, who captures the LVR is not important, but it is the block producer who must repay the LVR of a block. To enforce this, for a Diamond pool, we check the pool state in the first pool transaction each block and take escrow from the block producer. This escrow is be used in part to pay the realized LVR of the block back to the pool. The first pool transaction also returns the collateral of the previous block producer, minus the realized LVR (computable from the difference between the current pool state and the pool state at the beginning of the previous block). To ensure the collateral covers realized LVR, each proceeding pool transaction verifies that the LVR implied by the pool state as a result of the transaction can be repaid by the deposited collateral. Our implementation is based on the following two assumptions:

1. A block producer always sets the final state of a pool to the state which maximizes the LVR.
2. The block producer realizes net profits of at least the LVR corresponding to the final state of the pool.

If the final price of the block is not the price maximizing LVR, the block producer has ignored an arbitrage opportunity. The block producer can always ignore non-block producer transactions to realize the LVR, therefore, any additional included transactions must result in greater or equal utility for the block producer than the LVR.

5.1 Core Protocol

The first transaction interacting with a Diamond pool Φ in every block attests to the maximum and minimum prices attained by Φ during the block, P_{max} and P_{min} respectively. We call this transaction the *pool unlock transaction*. Only one pool unlock transaction is executed per pool per block. Given a current pool price of P_0 corresponding to reserves $(R_{x,0}, R_{y,0})$, it must be that $P_{min} \leq P_0 \leq P_{max}$. As such, given a move to P_{min}, some amount $\lambda_x \geq 0$ must be returned to the Φ pool and vault by the block producer. Similarly, given a move to P_{max}, $\lambda_y \geq 0$ must be returned to the Φ pool and vault by the block producer. For a final pool price P_1 with $\Delta_x > 0$ tokens added to the pool and $\Delta_y > 0$ tokens removed (implying $P_1 > P_0$), $\alpha\Delta_x$ tokens are removed from λ_x, while $\alpha\Delta_y$ tokens are returned to the producer who deposited λ_x. A further δ_x tokens are removed from λ_x such that:

$$P(R_{x,0} - (1-\alpha)\Delta_x - \delta_x, R_{y,0} + (1-\alpha)\Delta_y) = P_1. \tag{6}$$

The λ_y and remainder of λ_x, if any, are returned to the producer who deposited λ_x and λ_y. Given $P_1 \leq P_0$, the same process is repeated with $\alpha\Delta_y$ and δ_y paid to the pool and vault respectively. These amounts (λ_x, λ_y) must be deposited to the protocol contract in the pool unlock transaction.

Every proceeding user transaction interacting with Φ in the block first verifies that the implied pool move stays within the bounds $[P_{min}, P_{max}]$ specified at the start of the block. Non pool-unlock transactions are executed as they would be in the corresponding CFMM pool $CFMM(\Phi)$ **without** an α discount on the amount of tokens that can be removed. If a transaction implies a move outside of these bounds, it is not executed.

The next time a pool unlock transaction is submitted (in a proceeding block), given the final price of the preceding block was P_1 the actual amount of token x or y required to be added to the pool and vault (the $\alpha\Delta$ and δ of the required token, as derived earlier in the section) is taken from the deposited escrow, with the remainder returned to the block producer who deposited those tokens.

Remark 1. Setting the LVR-rebate parameter too high can result in protocol censorship and/or liveness issues as certain block producers may not be equipped to frictionlessly arbitrage, and as such, repay the implied LVR to the protocol. To counteract this, the LVR rebate parameter should be reduced every block in which no transactions take place. As arbitrageurs are competing to extract LVR from the pool, the LVR rebate parameter will eventually become low enough for block producers to include Diamond transactions.

5.2 Conversion Protocols

The described implementations in this section assume the existence of a decentralized on-chain auction. [7]

Per-block Conversion vs. Futures Given per-block conversion (Section 4.3), further deposits from the block producer are required to cover the token requirements of the conversion and collateralizing the futures. The conversions for a pool Φ resulting from transactions in a block take place in the next block a pool unlock transaction for Φ is called. Given a maximum expected percentage move over T blocks of σ_T, and a conversion of λ_y tokens at price P, the block producer collateral must be in quantities π_x and π_y such that if the block producer is long the futures:

$$1.\ \pi_x + \pi_y \frac{P}{1+\sigma_T} \geq \lambda_y (P - \frac{P}{1+\sigma_T}),\ \text{and}\ 2.\ \frac{\pi_x}{\pi_y} = \frac{P}{1+\sigma_T}. \tag{7}$$

If the block producer is short the futures it must be that:

$$1.\ \pi_x + \pi_y \frac{P}{1+\sigma_T} \geq \lambda_y P \sigma_T, \qquad \text{and}\ 2.\ \frac{\pi_x}{\pi_y} = P(1+\sigma_T). \tag{8}$$

[7] First-price sealed-bid auctions can be implemented using a commit-reveal protocol. An example of such a protocol involves bidders hashing bids, committing these to the blockchain along with an over-collaterlization of the bid, with bids revealed when all bids have been committed.

The first requirement in both statements is for the block producer's collateral to be worth more than the maximum expected loss. The second requirement states the collateral must be in the ratio of the pool for the maximum expected loss (which also ensures it is in the ratio of the pool for any other loss less than the maximum expected loss). This second requirement ensures the collateral can be added back into the pool when the futures are settled.

At settlement, if the futures settle in-the-money for the block producer, tokens are removed from the pool in the ratio specified by the settlement price with total value equal to the loss incurred by the pool, and paid to the block producer. If the futures settle out-of-the-money, tokens are added to the pool from the block producer's collateral in the ratio specified by the settlement price with total value equal to the loss incurred by the block producer. The remaining collateral is returned to the block producer. The pool constant is adjusted to reflect the new balances.

Remark 2. As converting the vault does not affect pool availability, the auctions for converting the vault can be run sufficiently slowly so as to eliminate the risk of block producer censorship of the auction. We choose to not remove tokens from the pool to collateralize the futures as this reduces the available liquidity within the pool, which we see as an unnecessary reduction in benefit to users (which would likely translate to lower transaction fee revenue for the pool). For high volatility token pairs, T should be chosen sufficiently small so as to not to risk pool liquidation.

If Diamond with conversion versus futures is run on a blockchain where the block producer is able to produce multiple blocks consecutively, this can have an adverse effect on incentives. Every time the vault is converted and tokens are re-added to the pool, the liquidity of the pool increases. A block producer with control over multiple blocks can move the pool price some of the way towards the maximal LVR price, convert the vault tokens (which has 0 expectancy from Lemma 2), increase the liquidity of the pool, then move the pool towards the maximal LVR price again in the proceeding block. This process results in a slight increase in value being extracted from the pool in expectancy compared to moving the pool price immediately to the price corresponding to maximal LVR. Although the effect on incentives is small, re-adding tokens from a conversion slowly/keeping the pool constant constant mitigates/removes this benefit for such block producers.

Periodic Conversion Auction Every T blocks, given θ tokens in the vault, $\frac{\theta}{2}$ of these tokens are auctioned off, with bids placed in the other token. The winning bidder receives these $\frac{\theta}{2}$. The winning bid, and the remaining $\frac{\theta}{2}$ tokens in the vault are added to the pool.

6 Experimental Analysis

This section presents the results of several experiments, which can be reproduced using the following anonymized repository [20]. The results provide further evidence of the performance potential of a Diamond pool versus various

benchmarks. These experiments isolate the effect that different fees, conversion frequencies, daily price moves, LVR rebate parameters, and days in operation have on a Diamond pool. Each graph represents a series of random-walk simulations which were run, unless otherwise stated, with base parameters of:

- LVR rebate parameter: 0.95.
- Average daily price move: 5%.
- Conversion frequency: Once per day.
- Blocks per day: 10.
- Days per simulation: 365.
- Number of simulations per variable: 500.

Each graph plots the final value of the Diamond Periodic Conversion Auction pool (unless otherwise stated) relative to the final value of the corresponding Uniswap V2 pool. The starting reserve values are $\$100m$ USDC and $76,336$ ETH, for an ETH price of $\$1,310$, the approximate price and pool size of the Uniswap ETH/USDC pool at the time of writing [22].

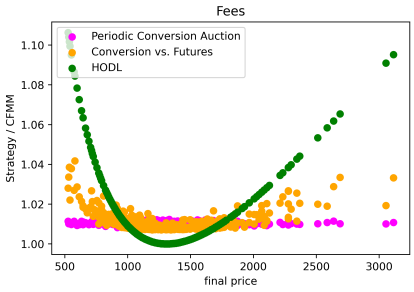

Fig. 3

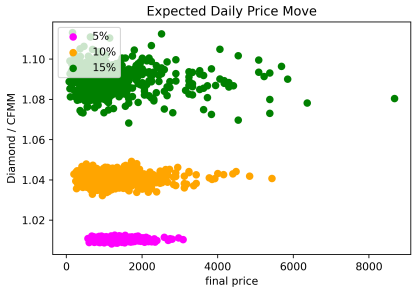

Fig. 4

Figure 3 compares four strategies over the same random walks. Periodic Conversion Auction and Conversion vs. Futures replicate the Diamond protocol given the respective conversion strategies (see Section 4). HODL (Hold-On-for-Dear-Life), measures the performance of holding the starting reserves until the end of the simulation. The final pool value of these three strategies are then taken as a fraction of the corresponding CFMM pool following that same random walk. Immediately we can see all three of these strategies outperform the CFMM strategy in all simulations (as a fraction of the CFMM pool value, all other strategies are greater than 1), except at the initial price of 1310, where HODL and CFMM are equal, as expected.

The Diamond pools outperform HODL in a range around the starting price, as Diamond pools initially retain the tokens increasing in value (selling them eventually), which performs better than HODL when the price reverts. HODL performs better in tail scenarios as all other protocols consistently sell the token increasing in value on these paths. Note Periodic Conversion slightly outperforms Conversion vs. Futures when finishing close to the initial price, while slightly underperforming at the tails. This is because of the futures exposure. Although

these futures have no expectancy for the protocol, they increase the variance of the Conversion vs. Futures strategy, outperforming when price changes have momentum, while underperforming when price changes revert.

Figure 4 identifies a positive relationship between the volatility of the price and the out-performance of the Diamond pool over its corresponding CFMM pool. This is in line with the results of [17] where it is proved LVR grows quadratically in volatility. Figure 5 demonstrates that, as expected, a higher LVR-rebate parameter α retains more value for the Diamond pool.

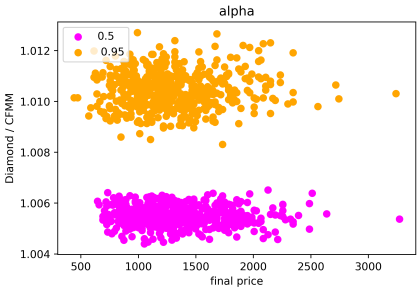

Fig. 5

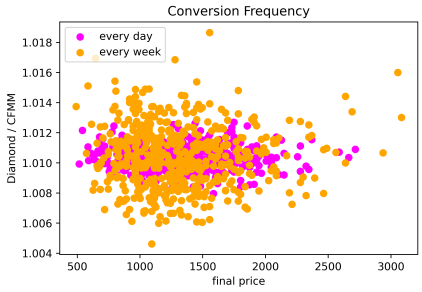

Fig. 6

Figure 6 shows that higher conversion frequency (1 day) has less variance for the pool value (in this experiment once per day conversion has mean 1.011234 and standard deviation 0.000776 while once per week conversion has mean 1.011210 and standard deviation 0.002233). This highlights an important trade-off for protocol deployment and LPs. Although lower variance corresponding to more frequent conversion auctions is desirable, more frequent auctions may centralize the players participating in the auctions due to technology requirements. This would weaken the competition guarantees needed to ensure that the auction settles at the true price in expectancy.

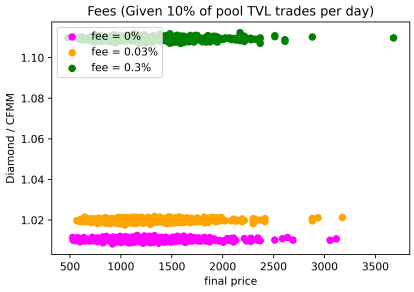

Fig. 7

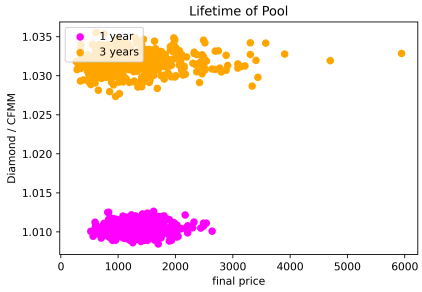

Fig. 8

Figure 7 compares Diamond to the CFMM pool under the specified fee structures (data-points corresponding to a particular fee apply the fee to both the Uniswap pool and the Diamond pool) assuming 10% of the total value locked in each pool trades daily. The compounding effect of Diamond's LVR rebates

with the fee income every block result in a significant out-performance of the Diamond protocol as fees increase. This observation implies that given the LVR protection provided by Diamond, protocol fees can be reduced significantly for users, providing a further catalyst for a DeFi revival. Figure 8 demonstrates that the longer Diamond is run, the greater the out-performance of the Diamond pool versus its corresponding CFMM pool.

7 Conclusion

We present Diamond, an AMM protocol which provably protects against LVR. The described implementation of Diamond stands as a generic template to address LVR in any CFMM. The experimental results of Section 6 provide strong evidence in support of the LVR protection of Diamond, complementing the formal results of Section 4.5. It is likely that block producers will be required to charge certain users more transaction fees to participate in Diamond pools to compensate for this LVR rebate, with informed users being charged more for block inclusion than uninformed users. As some or all of these proceeds are paid to the pool with these proceeds coming from informed users, we see this as a desirable outcome.

Given the protocol-level protections provided by Diamond, research must now focus on user protections in order to keep protocol utility in the hands of protocol contributors. Rook [19] and CoW protocol [9] have taken important steps in this regard, although many open problems still exist.

References

1. Adams, H., Keefer, R., Salem, M., Zinsmeister, N., Robinson, D.: Uniswap V3 Core (2021), `https://uniswap.org/whitepaper-v3.pdf`
2. Adams, H., Zinsmeister, N., Robinson, D.: Uniswap V2 Core (2020), `https://uniswap.org/whitepaper.pdf`
3. Bartoletti, M., Chiang, J.H.y., Lluch-Lafuente, A.: A Theory of Automated Market Makers in DeFi. In: Damiani, F., Dardha, O. (eds.) Coordination Models and Languages. pp. 168–187. Springer International Publishing, Cham (2021)
4. Bartoletti, M., Chiang, J.H.y., Lluch-Lafuente, A.: Maximizing Extractable Value from Automated Market Makers. In: Financial Cryptography and Data Security. Springer Berlin Heidelberg, Berlin, Heidelberg (2022)
5. Bichuch, M., Feinstein, Z.: Axioms for Automated Market Makers: A Mathematical Framework in FinTech and Decentralized Finance (2022), `https://arxiv.org/abs/2210.01227`
6. Budish, E., Cramton, P., Shim, J.: The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response *. The Quarterly Journal of Economics **130**(4), 1547–1621 (07 2015). https://doi.org/10.1093/qje/qjv027, `https://doi.org/10.1093/qje/qjv027`
7. Capponi, A., Jia, R.: The adoption of blockchain-based decentralized exchanges (2021), `https://arxiv.org/abs/2103.08842`
8. Chainlink: `https://chain.link/`

9. CoW Protocol: https://docs.cow.fi/
10. Evans, A., Angeris, G., Chitra, T.: Optimal fees for geometric mean market makers. In: Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S. (eds.) Financial Cryptography and Data Security. FC 2021 International Workshops. pp. 65–79. Springer Berlin Heidelberg, Berlin, Heidelberg (2021)
11. @hagaetc: `https://dune.com/queries/4494`
12. Josojo: Mev capturing amms (2022), `https://ethresear.ch/t/mev-capturing-amm-mcamm/13336`
13. Krishna, V.: Auction theory. Academic press (2009)
14. Krishnamachari, B., Feng, Q., Grippo, E.: Dynamic automated market makers for decentralized cryptocurrency exchange. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–2 (2021). https://doi.org/10.1109/ICBC51069.2021.9461100
15. Loesch, S., Hindman, N., Richardson, M.B., Welch, N.: Impermanent Loss in Uniswap v3 (2021), `https://arxiv.org/abs/2111.09192`
16. McMenamin, C., Daza, V., Fitzi, M., O'Donoghue, P.: Fairtradex: A decentralised exchange preventing value extraction. In: Zhang, F., McCorry, P. (eds.) Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security. ACM (2022)
17. Milionis, J., Moallemi, C.C., Roughgarden, T., Zhang, A.L.: Quantifying loss in automated market makers. In: Zhang, F., McCorry, P. (eds.) Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security. ACM (2022)
18. Park, A.: The conceptual flaws of constant product automated market making. ERN: Other Microeconomics: General Equilibrium & Disequilibrium Models of Financial Markets (2021)
19. Rook: https://docs.rook.fi/reference/
20. Diamond simulation: `https://anonymous.4open.science/r/LVR-0D11/MainProtocolSimulation.py`
21. @thiccythot: `https://dune.com/thiccythot/uniswap-markouts`
22. Uniswap: `https://app.uniswap.org/`

A Proofs

Lemma 4. *Converting the vault every block vs. future contracts has expectancy of at least 0 for a Diamond pool.*

Proof. Consider a conversion of θ tokens which takes place at time 0. Let the conversion be done at some price p_c, while the external market price is p_0. WLOG let the protocol be selling θ y tokens in the conversion, and as such, buying θ y token futures at price P_c. The token sells have expectancy $\theta(P_c - p_0)$. For the strategy to have at least 0 expectancy, we need the futures settlement to have expectancy of at least $\theta(p_0 - P_c)$. In Section 4.3, two versions of this strategy were outlined. We consider both here. In both sub-proofs, we use the assumption that the risk-free rate is 0, it must be that the external market price at time t is such that $\mathbb{E}(p_t) = p_0$. We now consider the two options for settling futures outlined in Section 4.3

Option 1: Settle futures by auctioning tokens at the original converted price. The arbitrageur who converted tokens for the pool at price P_c

must auction off the tokens at price P_c. Let the auction happen at time t, with external market price at that time of p_t. Notice that what is actually being sold is the right, and obligation, to buy θ tokens at price P_c. This has value $\theta(p_t - P_c)$, which can be negative. As negative bids are paid to the auction winner by the protocol, and positive bids are paid to the protocol, we are able to apply Lemma 1. As such, the winning bid is at least $\theta(p_t - P_c)$, which has expectancy of at least

$$\mathbb{E}(\theta(p_t - P_c)) = \theta(\mathbb{E}(p_t) - P_c) = \theta(p_0 - P_c). \tag{9}$$

Thus the expectancy of owning the future for the protocol is at least $\theta(p_c - P_0)$, as required.

Option 2: Settle futures using frequent batch auction settlement price. For a swap with external market price p_t at time t, a batch auction in this swap settles at p_t in expectancy (see Theorem 5.2 of [16]). Thus the futures owned by the protocol have expectancy

$$\mathbb{E}(\theta(p_t - P_c)) = \theta(\mathbb{E}(p_t) - P_c) = \theta(p_0 - P_c). \tag{10}$$

Lemma 5. *A periodic conversion auction has expectancy of at least 0 for a Diamond pool.*

Proof. Consider a Diamond pool Φ with vault containing 2θ tokens. WLOG let these be of token y. Therefore the pool must sell θ tokens at the external market price to balance the vault. Let the conversion auction accept bids at time t, at which point the external market price is p_t. For the auction to have expectancy of at least 0, we require the winning bid to be at least θp_t. The result follows from Lemma 1.

Theorem 1. *For a CFMM pool $CFMM(\Phi)$ with LVR of $L > 0$, the LVR of Φ, the corresponding pool in Diamond, has expectancy of at most $(1 - \alpha)L$.*

Proof. To see this, we first know that for $CFMM(\Phi)$ at time t with reserves $(R_{x,t}, R_{y,t})$, the optimal solution to the pool value function with external market price p_{t+1} corresponds to updated reserve values $(R^*_{x,t+1}, R^*_{y,t+1})$ which minimize:

$$(R_{x,t} - R_{x,t+1}) + (R_{y,t} - R_{y,t+1})p_{t+1}. \tag{11}$$

Let this quantity be

$$L = (R_{x,t} - R^*_{x,t+1}) + (R_{y,t} - R^*_{y,t+1})p_{t+1}. \tag{12}$$

In Diamond, a player trying to move the reserves of Φ to $(R'_{x,t+1}, R'_{y,t+1})$ only receives $(1 - \alpha)(R'_{x,t+1} - R_{x,t})$ while giving $(1 - \alpha)(R'_{y,t+1} - R_{y,t})$ to Φ. Thus, an arbitrageur wants to find the values of $(R'_{x,t+1}, R'_{y,t+1})$ that maximize:

$$(1 - \alpha)(R'_{x,t+1} - R_{x,t}) + (1 - \alpha)(R'_{y,t+1} - R_{y,t})p_{t+1} + \mathbb{E}(\text{conversion}). \tag{13}$$

where $\mathbb{E}(\text{conversion})$ is the per-block amortized expectancy of the conversion operation for the arbitrageurs. From Lemma 1, we know $\mathbb{E}(\text{conversion}) \geq 0$

for the Φ. Therefore, the LVR of Φ corresponds to the values $(R'_{x,t+1}, R'_{y,t+1})$ minimizing the function:

$$(1-\alpha)\big((R_{x,t} - R'_{x,t+1}) + (R_{y,t} - R'_{y,t+1})p_{t+1}\big) \tag{14}$$

which is just the negative of Equation 13. From Equation 12, we know this has a minimum at $(R'_{x,t+1}, R'_{y,t+1}) = (R^*_{x,t+1}, R^*_{y,t+1})$. Therefore, the LVR of Φ is at most:

$$(1-\alpha)((R_{x,t} - R^*_{x,t+1}) + (R_{y,t} - R^*_{y,t+1})p_{t+1}) = (1-\alpha)L. \tag{15}$$