An AMM minimizing user-level extractable value and loss-versus-rebalancing

Conor McMenamin & Vanesa Daza

Supported by EU H2020 BAnDIT Project 814284 & AEI-PID2021-128521OB-I00 grant of Spanish Ministry of Sci. & Ed.
Motivation

DEX protocol liquidity and order inclusion are typically controlled *exclusively* by the block producer.

This monopoly is particularly profitable when:

1. The liquidity is stale, not updating to current information: *loss-versus-rebalancing (LVR)* (Millionis et al., 2022).
2. The orders are unencrypted: *front-/back-running, sandwiching*.

We provide **VOLVER**, an AMM protocol addressing both of these sources.
Techniques:

- Encrypt orders.
  - V0LVER orders are allocated while encrypted.
  - No information revealed pre-allocation.
- Update liquidity.
  - Single execution price.
  - Producer must provide some $\beta \in [0,1]$ of liquidity to allocated orders.
  - Incentivized to allocate liquidity at external market price.
Decentralized Exchange Losses

Cumulative Extracted MEV - Gross Profit

$687M
LVR Protection in V0LVER

Orders cannot be executed until block producer updates the implied pool price.

These updates are typical buy/sell orders, with 2 caveats:

1. Some percentage $\beta \in [0,1]$ of an update order is not executed.
   a. Pool price reflects the implied move of the original update order (before $\beta$ is applied).
   b. Excess pool tokens are added to a vault.
2. The producer must attest to this pool price.

Attesting to a pool price: If $n$ orders are allocated after an update tx, the producer must provide $\beta$ of the liquidity for those orders.
Excess Pool Tokens

Consider Uniswap V2, where for reserves $R_x$, $R_y$ the implied price is $R_x/R_y$, and reserves updated according to $R_x \cdot R_y = K$, the pool constant.

In Uniswap V2 (and VOLVER), optimal producer update is move implied price to external market price. (check!)

If only $\beta$ of order is executed, we need to remove pool tokens to ensure implied price equals external market price.
Attest to Price, Provide $\beta$ of Liquidity

The $n$ orders are batch executed, equiv. to one meta order.

Meta order executed according to the pool invariant function ($R_x \cdot R_y = K$ for Uni V2) at the attested price/reserve ratio.

$\beta$ of the sent/received tokens are received/sent by the block producer.

As $n$ and max order size are known when submitting update order, the max necessary liquidity is allocated from pool: block producer in a ratio of $(1-\beta): \beta$. 
Non-LVR MEV

As mentioned, liquidity is allocated to orders.

Allocated orders in V0LVER are encrypted, must be decrypted to be executed.

Depending on the encryption used (threshold/committee controlled vs. user controlled), decryption may occur in the next block, or later.

Decryption must be incentivized (not decrypting punished).

Similarly, we must hide user-/order-information until order is allocated.

ZK commitment schemes allow for this.
Putting it all together

Under producer competition, block producers compete to allocate orders and submit update transactions.

In V0LVER, this keeps $\beta$ high, which:

Effectively eliminates LVR (update tx extracts $\beta$ of LVR).

Even under producer monopoly, as long as orders are encrypted when allocated:

Users trade at external market price, in expectancy, minus impact and fees.
Questions?

Twitter: @ConorMcMenamin9
Email: conor.mcmenamin@upf.edu

Arxiv version: https://arxiv.org/abs/2301.13599
Graphical Representations of V0LVER
Figure 2: Flow of V0LVER protocol, excluding the allocation protocol (see Figure 3 for the allocation protocol). The double-border rectangle is the initialization state, thin single-border rectangles are state updates on-chain, while thick-bordered rectangles are block producer decisions/computations off-chain. The circle state is controlled by the network. Note that \( In \), the array of inserted but unallocated OCTs, is an ordered array of sets of OCTs. For \( 1 < a \leq len(In) \), \( In[:a] \) returns an ordered sub-array of \( In \) elements at indices \([1, \ldots, a]\), while \( In[a:] \) returns an ordered sub-array of \( In \) elements at indices \([a, \ldots, len(In)]\).
Figure 3: Flow of allocation protocol for V0LVER pool $\Phi$, initialized every time the ALLOCATE() function is called in Figure 2. The Reveal Orders state happens by some block after height $H$. As in the previous figure, the double-border rectangle is the initialization state, thin single-border rectangles are state updates on-chain, while thick-bordered rectangles are block producer decisions/computations off-chain.