Front-running on AMMs is a big (and expensive) problem. It causes lost money during large swaps and also raises gas prices unnecessarily

due to bidding wars between bots.

I propose a simple method to mitigate AMM front-running by applying zero-knowledge proofs (ZKPs) for updating the swap ratio.

Abstract

Each swap transaction on an AMM can “hide” the updated pool ratio (and therefore resulting price) by means of a SNARK puzzle, thereby making price-prediction of a transaction in the mempool expensive and uncertain, significantly deterring front-running.

Scenario

Let’s assume that we have a Uniswap-like pool for the ETH/BTC pair. The mechanism is pretty simple.

1. A trader executes a swap and updates the pool ratio but

hides the details in a SNARK that proves the AMM mathematics remain consistent.

1. The trade transaction contains a hint that makes it possible to guess the ratio by n

times of hash computation (where n

should not be too large to prevent falling into limbo). So anyone can compute the updated pool ratio by solving the SNARK puzzle after n

computations.

1. Front-runners will fail to predict the price change easily and will be deterred from making a sandwich attack due to uncertainty.

And here we can add some tokenomics to run the system more efficiently.

1. The trader can decide the SNARK puzzle difficulty n

in the swap transaction.

1. The trader should stake x

amount of tokens, where x

should be proportional to n

.

1. After the swap, one of two things can happen:

a. If the trader reveals

within 10 minutes, the staked tokens go back to the trader.

b. If the trader does not reveal

within 10 minutes, anyone can withdraw the staked tokens by submitting the SNARK puzzle answer.

Difficulty Fee

Since increasing puzzle difficulty (n

) protects the trader while imposing a potential cost on others, we can also implement a “difficulty fee”, taken from the staked tokens x

if n

is beyond a certain threshold.

Implementation

With this idea, I designed an anti-front-running AMM protocol “Snarkswap” based on Uniswap math. You can see the detail

protocol design here, and the implementation here.

I need your feedback

Please let me know what you think of this idea, thank you!