Abstract

Recent work [0] has demonstrated that if a consensus algorithm directly interacts with a constant function market maker (CFMM) like Uniswap, then DEX users can have partial privacy for their trades. This post will elucidate how this mechanism works and provide some background. Note that this mechanism relies on a verifiable random function being present to guarantee a high cost of manipulation for trade ordering MEV. Given that ETH2 has RANDAO, I wanted to start a discussion on whether base protocols should provide private access to DEXs to users as a base protocol primitive.

Background

Privacy in decentralized exchanges is a nascent research field. However, as volumes bloom, it has become much more salient to market participants that private DEX transactions are extremely difficult to implement. Prior work on this topic was inspired in part by @barryWhiteHat's ethresearch post "Why you can't build a private Uniswap with ZKPs". In particular, our paper [2] proved that any constant function market maker (CFMMs) such as Balancer, Curve, or Uniswap, cannot achieve privacy strictly from blinding reserves/liquidity. However, there are two ways out of this impossibility result:

1. Batching
2. Adding noise to prices/trade sizes

Both of these options will cause worse execution prices for traders, but will increase transaction size privacy. There are a few things to note:

- The noise added to prices needs to not be independent of the trades submitted. For instance, if one adds \xi_i \sim \mathsf{N}(0, \sigma^2)

noise to each trade size and/or price, then with a large enough set of trades, it is possible to denoise trade sizes.

- Batching works well when the distribution of trade sizes is concentrated, however, it works quite poorly when there is one trade whose size is much large than the rest, e.g. a sequence of T

trades (T, 1, \ldots 1) \in \mathbb{R}^T

. This is because the execution price for the small trades of size 1 is worsened dramatically by the trade of size T

.

- Similarly, a lack of batching would allow users to extract MEV from the "whale" trade of size T

. Suppose an adversary has a sequence of binary classifier B_i : \mathbb{R}^i \rightarrow {0, 1}

such that B_i(T_1, \ldots, T_i)

is 1 if the whale trade is in {T_1, \ldots, T_i}

and the probability that B_i

is accurate for all i

is greater than \frac{1}{2}+\gamma

. Then the adversary has 1-O(\gamma^T)

probability of figuring out the whale trade (this uses standard max margin / boosting arguments; c.f. Chapter 3 of [Boosting: Foundations and Algorithms

](https://mitpress.mit.edu/books/boosting)). If the adversary can predict the whale trade position, they can front run the whale trade by adding a trade T^{in}_{adv}

right before the whale trade and then another trade T^{out}_{adv}

after the whale trade. This is effectively a probabilistic sandwich attack.

The last note suggests that in order to achieve privacy against an adaptive adversary (e.g. they have access to binary classifiers with advantage \gamma

) one needs to be able to randomly permute input trades before execution. It also suggests that CFMM MEV (e.g. online learnability of trade sizes) is intrinsically tied to any realizable notion of privacy. Many recent research papers [5, 6] have demonstrated that the correct notion of privacy in online learning settings (which crypto folks would call adaptive adversaries) is that of differential privacy

.

What is differential privacy? Briefly speaking, a randomized algorithm \mathcal{A}

is (\epsilon, \delta)

-differentially private if

\mathsf{Pr}[\mathcal{A}(S) \in B] \leq e^{\epsilon} \mathsf{Pr}[\mathcal{A}(S') \in B] + \delta

for all input databases/datasets S, S'

with d(S, S') \leq 1

and all measurable B

. Usually, for discrete databases, the notion of distance used is the Hamming distance. This definition effectively is a sort of maximum KL-divergence bound — if you remove or add one data point to S

, then the information an adaptive adversary can learn is bounded by \epsilon

.

How does this apply to CFMMs? If we have a sequence of two sequence of trades T_1, \ldots, T_n

, T'*1, \ldots, T'*{m}

, a differentially private mechanism would make it impossible for an online learning adversary who has access to prices p_1, \ldots, p_n

and p'_1, \ldots, p'_m

to distinguish between the two sets of trades with probability dependent on \epsilon

and error dependent on \delta

. In particular, for different choices of \epsilon, \delta

, we can interpolate between the batched case and the MEV case.

Mechanism for Differential Privacy

In our paper [0], we demonstrate that one can achieve (\epsilon, \delta)

differential privacy by permuting, randomizing, and splitting trades. We will give some intuition here for why this works via examples with the hope that one can understand the mechanism without reading the paper.

The last section illustrated that randomly permuting trades is necessary for privacy against online learning adversaries. But is it sufficient? It turns out not to be sufficient, unfortunately. Let’s again consider the trade sequence \Delta = (T, 1, \ldots, 1) \in \mathbb{R}^T

. Draw a random permutation \pi \sim S_n

, where S_n

is the symmetric group on n

letters, and define \Delta^{\pi} = (\Delta_{\pi(1)}, \ldots, \Delta_{\pi(n)})

. For example if \pi = (1, 2)

, the transposition of the first and second elements, then \Delta^{\pi} = (1, T, 1, \ldots, 1)

. However, note that even though there are T!

permutations, there are only T

output trade sequences (the ones with T

in the i

th place for all i \in [T]

). This means that an adversary only has to learn a “threshold function” (e.g. the binary classifiers form the last section) to be successful at an MEV attack. This loss of entropy (from \log T! = O(T\log T)

to \log T

bits) means that an adaptive adversary will be exponentially more successful on this trade sequence than one where there is a bijection between \pi \leftrightarrow \Delta^{\pi}

.

On the other hand, consider a randomized trade vector \tilde{\Delta} = (T, 1 + \xi_i, \ldots, \ldots, 1 + \xi_T) \in \mathbb{R}^T

where \xi_i \sim \mathsf{N}(0, \sigma^2)

. Since \mathsf{Pr}[\tilde{\Delta}_i = \tilde{\Delta}_j] = 0

, randomly permuting this vector will lead to T!

different trade orders. As such, it is clear that we need to add some

noise to the actual trade sizes to achieve privacy. But is noise enough? Suppose that trades of size T

cause the price to change by at least \kappa T

for some \kappa > 0.

Then even if T!

trade sequences are unique, an adversary can attempt to look at price changes and filter them based on whether they are above or below a threshold dependent on \kappa

. Therefore, for privacy, we also need the trades executed to be “roughly of the same size” in order for this to be unsuccessful.

In [1], we show that splitting trades has a cost dependent on \kappa

(which is a measure of curvature

of an invariant function of a CFMM). Therefore, splitting trades to get them to be roughly the same size will have a price impact cost of roughly \kappa \log \max_i \Delta

. For the trade vector \Delta

, this is an exponential improvement in price impact when compared to raw batching (which has \Omega(T)

price impact vs \kappa \log T

impact). The main tools we use for getting this exponential improvement (at a cost of added randomness) involve mapping the permuted, noised, and split-up trades to a tree whose height / width control the privacy vs. price impact trade-off.

The formulas in [0] provide a mechanism for how to choose a certain amount of noise and trade splitting to achieve (\epsilon, \delta)

-differential privacy. This provides a direct utility (e.g. worsen price execution) vs. privacy trade-off for CFMMs like Uniswap as a function of their curvature (c.f. [1]).

Applicability to ETH2

I believe that smart contracts in ETH2 have access to a verifiable random function with some amount of entropy. The amount of entropy (roughly, up to small constant factors) required by this mechanism for n

trades is

- O(n \log n)

bits for the random permutation

- O\left(\frac{n}{\epsilon}\right)

bits for n

samples from a uniform distribution on [0, 1]

at \epsilon

precision [3]

Is this possible on ETH2? And another question: will smart contracts be able to access excess entropy produced by the VRF? I believe that Cosmos chains (such as Osmosis) allow for a CFMM to access the VRF directly but it requires the CFMM's execution to be more closely tied to consensus. Would such a mechanism need to be closely tied to consensus in ETH2? Or could it be implemented directly in a smart contract?

Open Problems

I would be remiss without providing some open problems that were opened up by this paper

- Local Differential Privacy

: Currently, the mechanism chooses the parameters for noise and trade splitting as a function of the whole set of trades \Delta \in \mathbb{R}^T

that are to be executed. Can we do something where users can add their own randomness (locally) and basically "pre-pay" for a certain amount of privacy? If this were true, one could remove the O\left(\frac{n}{\epsilon}\right)

bits of VRF entropy and allow users to choose their own individual utility vs. privacy trade-off.

- Improving Constants

: The constants that map ("amount of trade splitting", "amount of noise") to (\epsilon, \delta)

are not the best and likely can be improved by using improved smooth sensitivity [4]

- Implementation Details

: There are a lot of implementation details to sort out — how do we coarsely estimate curvature on-chain (easy for Uniswap, harder for Curve — see [1] for details) — and this will need numerical estimates for parameter choices.

Acknowledgements

Thanks go out to@g, Yi Sun, GaussianProcess, @ValarDragon, @kobigurk, @Pratyush

Main References

[0] Differential Privacy in Constant Function Market Makers, T. Chitra, G. Angeris, A. Evans, August 2021

[1] When does the tail wag the dog? Curvature and market making, G. Angeris, A. Evans, T. Chitra, December 2020

[2] A note on privacy in constant function market makers, G. Angeris, A. Evans, T. Chitra, March, 2021

[3] We need n

samples of the uniform distribution in order to construct a Dirichlet sample, e.g.,via stick-breaking

[4] Average-Case Averages: Private Algorithms for Smooth Sensitivity and Mean Estimation, M. Bun and T. Steinke, June 2019

[5] [Private PAC learning implies finite Littlestone

dimension, N. Alon, et. al, March 2019](http://www.cs.tau.ac.il/~nogaa/PDFS/adp.pdf)

[6] An Equivalence Between Private Classification and Online Prediction, Bun, et. al, June 2021