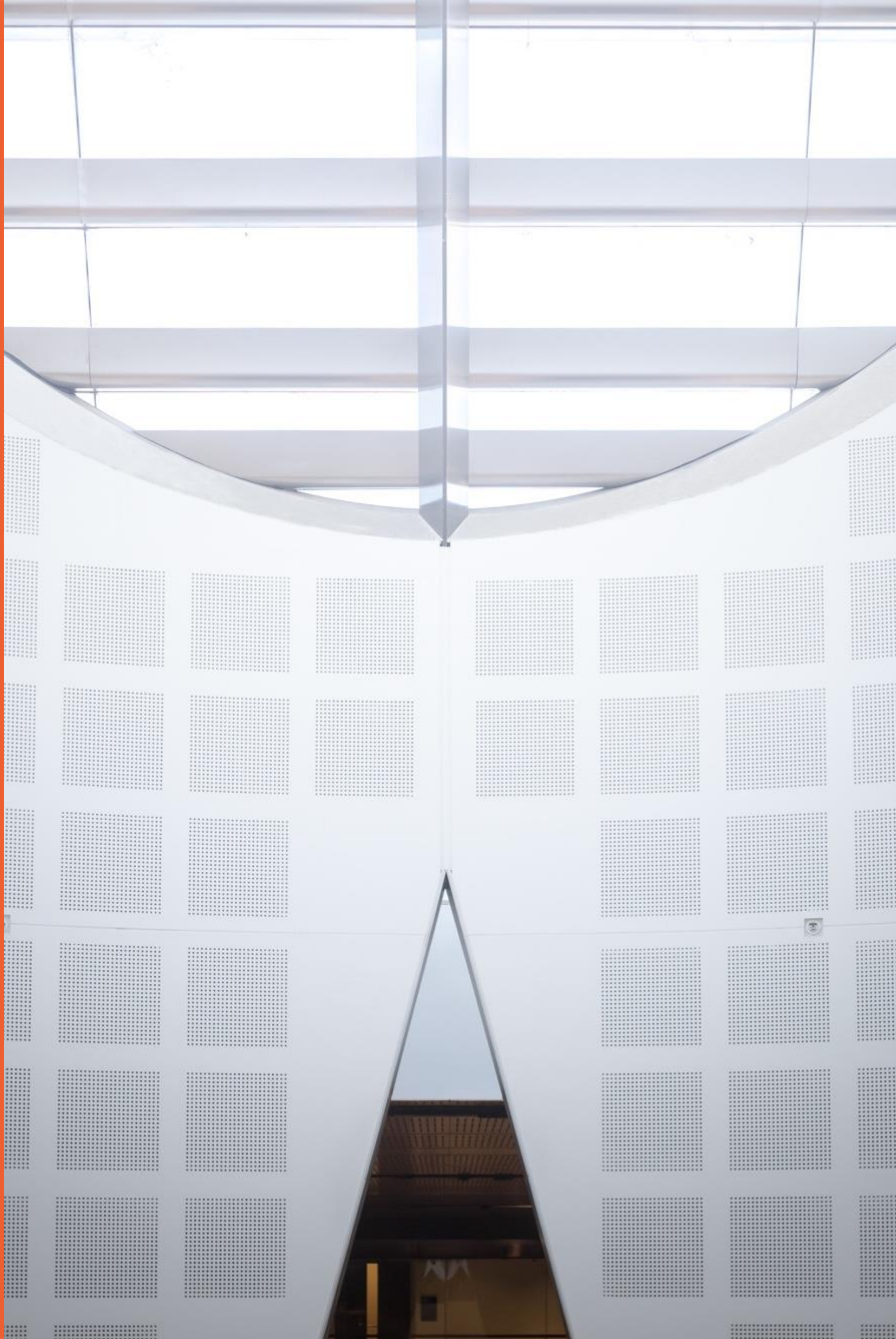
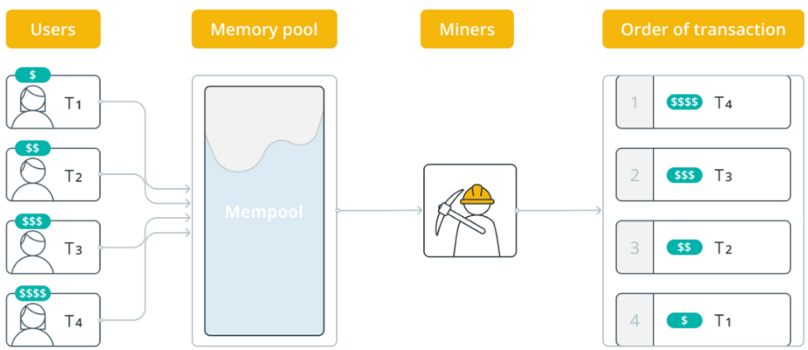
Evaluation of MEV Mitigation Technique

Presented by
Mohammad Saad Azam
Supervisor: Dr. Vincent Gramoli
School of Computer Science

THE UNIVERSITY OF SYDNEY



Motivation – MEV and its Impact



- **Untapped Resources**: MEV (Maximal Extractable Value) refers to value that can be extracted by reordering, inserting, or censoring transactions in a blockchain.

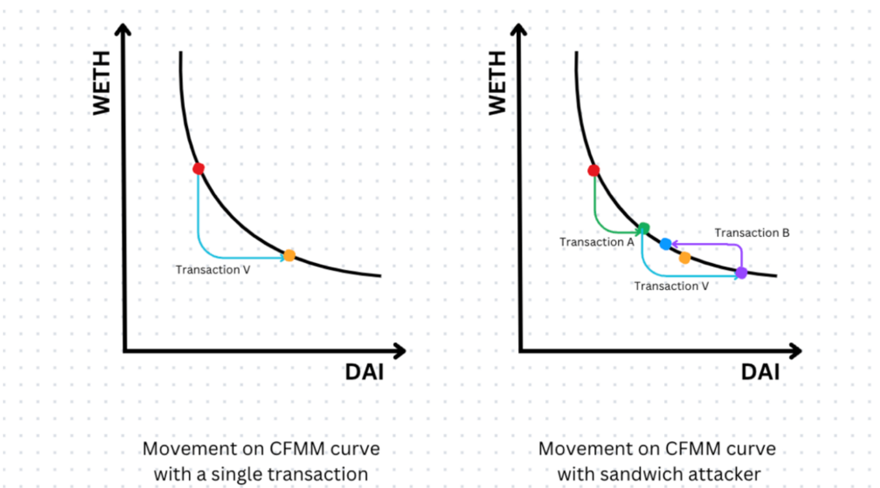
- **Manipulatable by Transaction Ordering**: The ability to control transaction sequencing allows validators to exploit opportunities, often at the expense of fairness and user profits.

Automated Market Makers

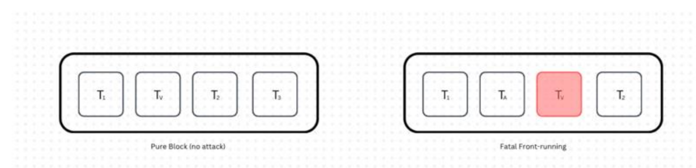
CFMM-Based DEXs: determine trade prices based on reserve ratios

Sandwich Attacks: exploit transaction positions in the pool, extract profits at the expense of users

Front-Running Attacks: exploit transaction ordering by placing trades ahead of user transactions, rendering them less effective or invalid



Sandwiching transactions



Front-running transactions

Background literature

MEV Exploitation:

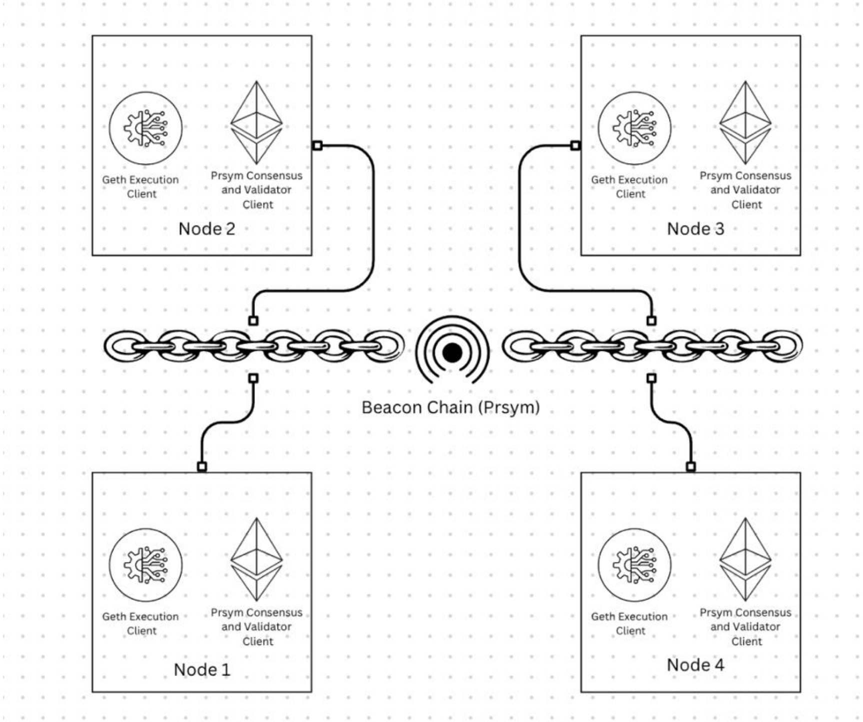
- Literature exploring MEV bots [FlashBoys2.0]
- Survey MEV attack surfaces [HowDarkIsTheForest]
- Formalize MEV extraction [Specter]
- Survey existing mitigation techniques [Preventing Transaction Reordering]

Mitigation Techniques:

- There exist several proposed mitigation techniques
- Some require change in protocols
  - Solutions like [Lyra], [Themis], [Pompeii]
  - Not all guard against a rushing adversary
- Others can be adapted into existing blockchains
  - Commit-and-Reveal schemes obfuscate transaction details
  - Time-based fairness consolidate ordering based on when the transactions arrive at a majority of nodes
  - Third party services which centralize the ordering, like Flashbots, Eden or OpenMEV
- Most proposed solutions are tested theoretically and not on real-world scenarios

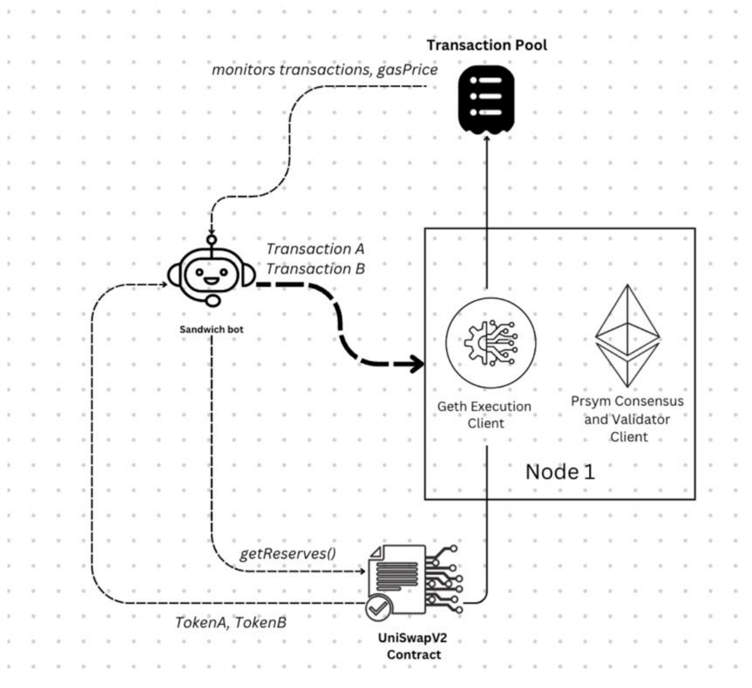
Methodology

- We evaluate a proposed mitigation technique, “Eating Sandwiches: Modular and lightweight elimination of transaction reordering attacks ”, in a real-world setting



Our test network consists of 4 full nodes, using Geth and Prysm

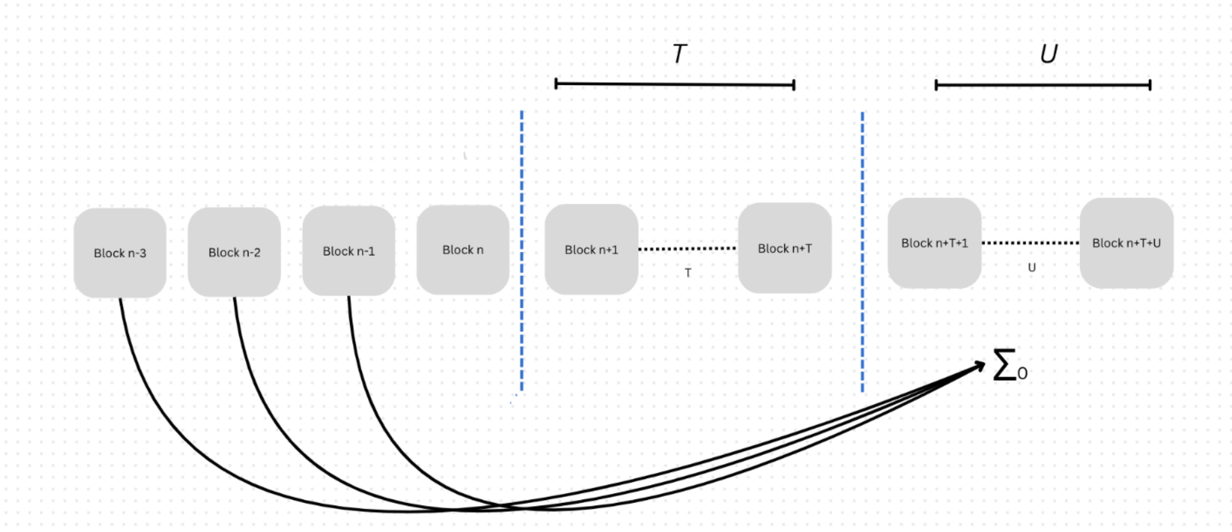
- We deploy UniSwap V2 contracts, and several other helping contracts



- We also deploy a custom sandwich bot to monitor the transaction pool and mount sandwiching attacks on viable swaps

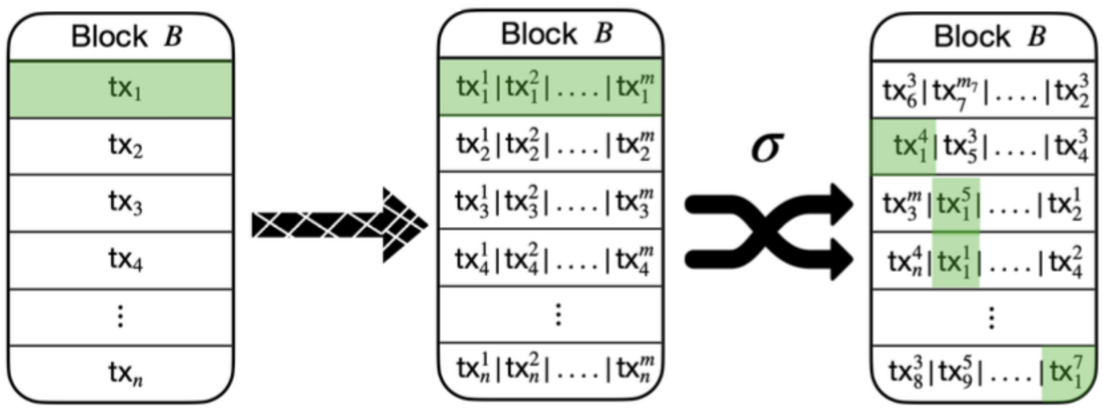
Implementation of protocol

- Protocol makes use leaders from previous blocks to generate partial seeds



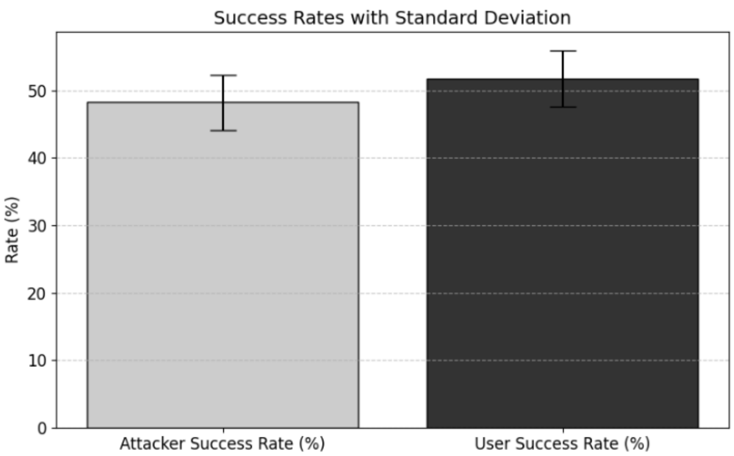
- Commitment to the seeds are stored and once the block is mined the ordering is determined with the new combined seed

- The transactions in the block are then chunked
- The chunked transactions are then permuted according to the random value

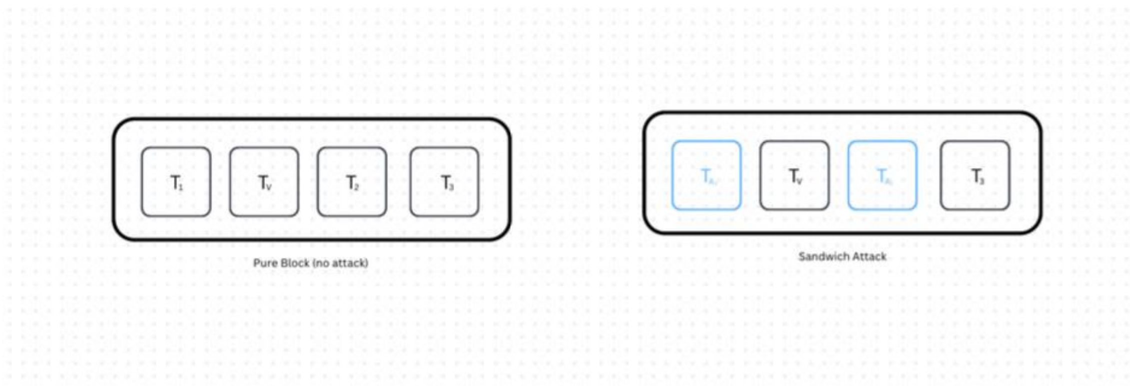


Results

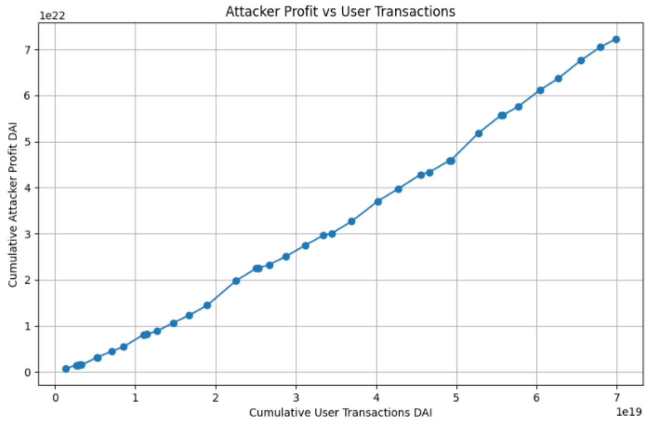
– Fatal Front-running:
- Deploy a custom contract to simulate an atomic smart contract
- Run the experiment on unmodified setup
- Repeat on modified setup

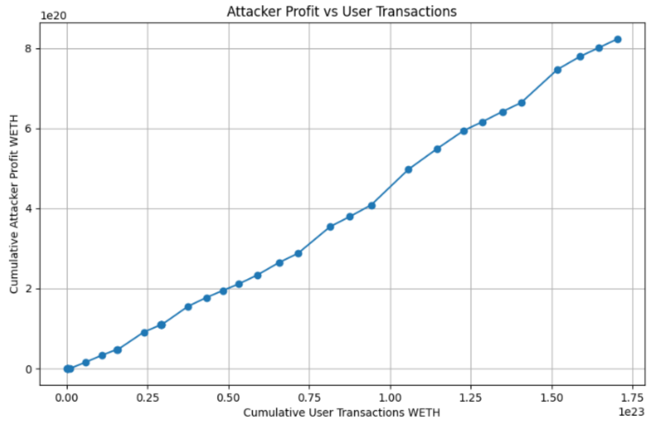


Results



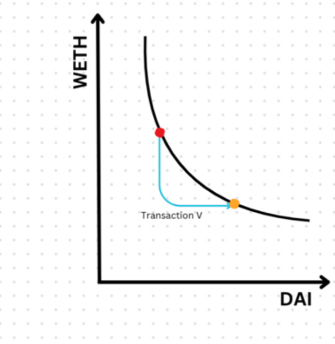
- Sandwich attack
- Simulate DAI/WETH Uniswap pool from 3 weeks of Etherscan data
- Repeat the test on vanilla setup and modified setup
- Vary the chunk size to evaluate effects of chunk size



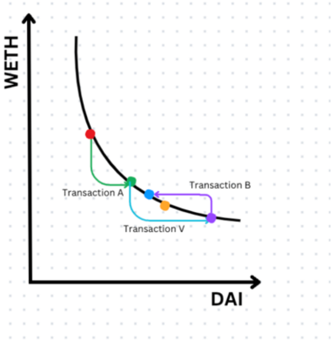


Default behavior (no mitigation)

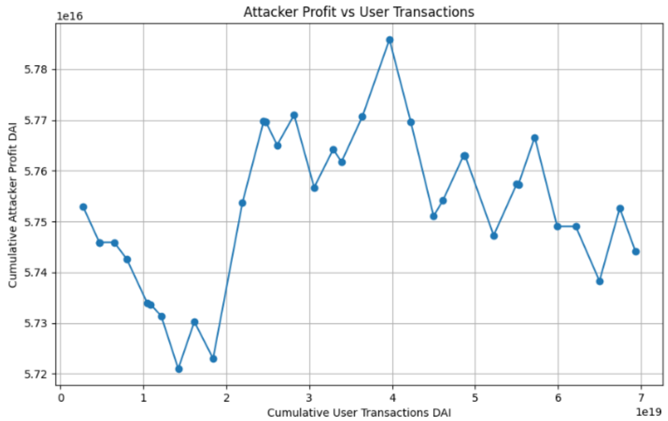
- Clear results indicating increasing profit with increasing user transaction amount; higher the amount greater the slippage

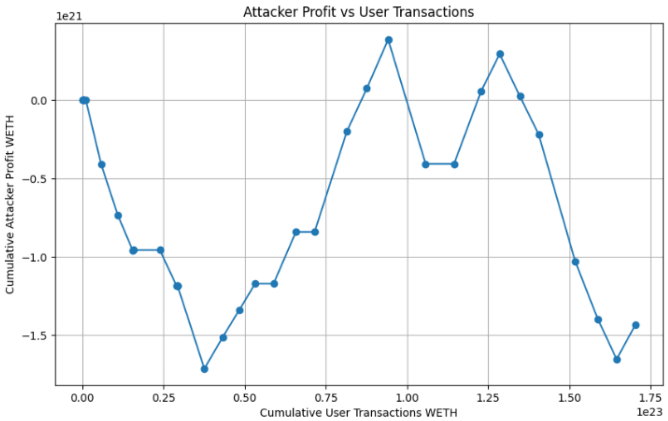


Movement on CFMM curve with a single transaction



Movement on CFMM curve with sandwich attacker

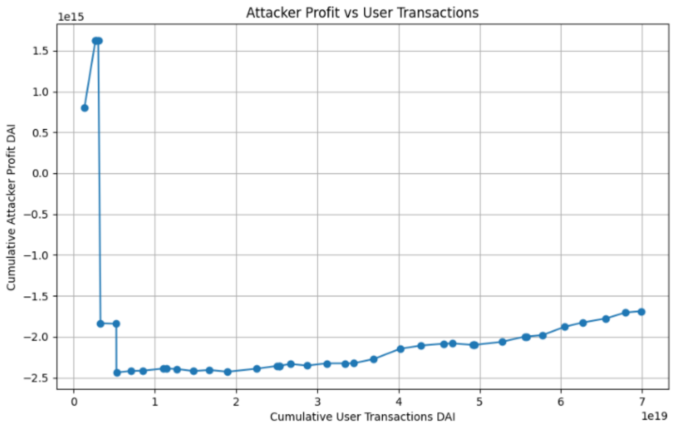


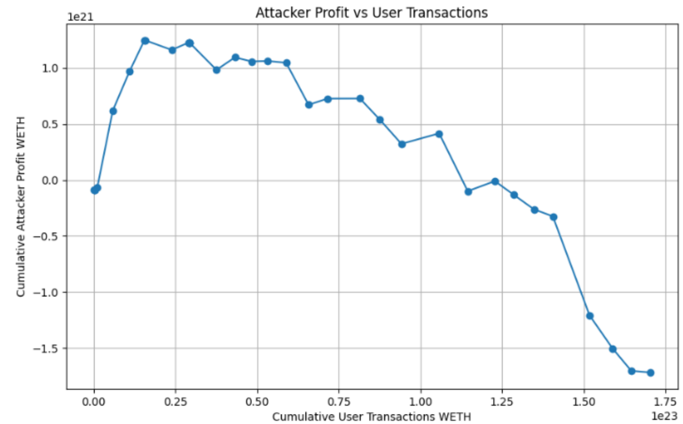


Reordering only (chunk size = 1)

- Results indicate effectiveness of reordering, attacker profits no longer guaranteed

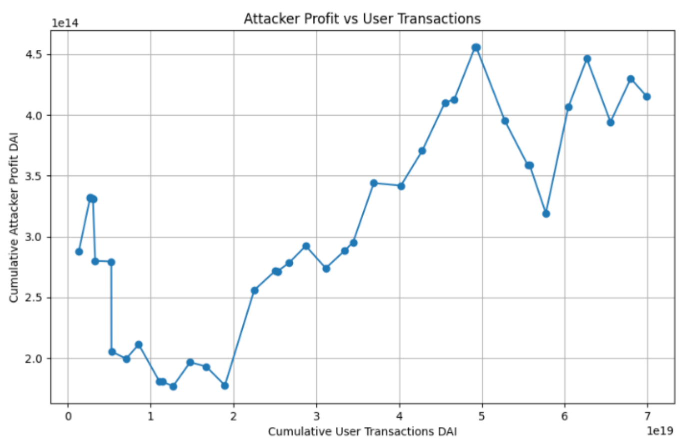
- Profit or loss vary in size
  - 1/6 likely to make profit
  - 1/6 likely to lose money (user makes profit)
  - 4/6 to either make 0, or some amount in between

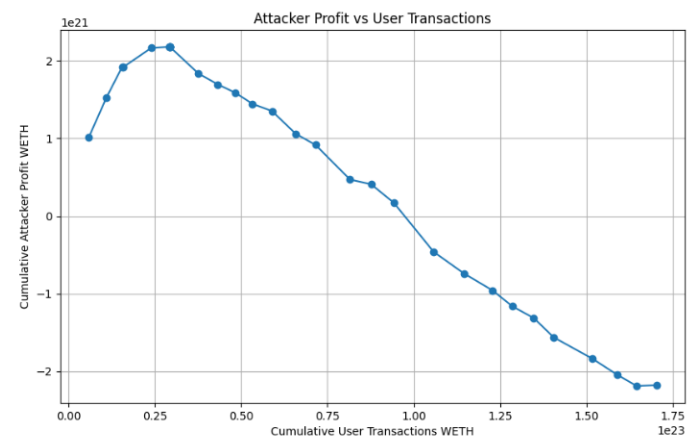




Reordering only (chunk size = 3)

- Results indicate effectiveness of chunking – changes in profit/loss become much smaller
- Dependent on original user transaction amount





Reordering only (chunk size = 6)

Discussion

Scope:

- Can address any sort of MEV attack, regardless of inner workings
- Transactions still made public hence attacker has 50% of success for fatal-front runs
  - Attacker will still launch attacks if expected return outweighs potential losses (gas fee if attack fails)
  - Compared to commit-reveal, this is a disadvantage
- Sandwich attacks (or attacks leveraging slippage) eliminated completely
  - Attacker as likely to make a profit or loss
  - Chunking reduces amount of profit/loss

Discussion

Jostling:

- Eliminates sandwich attacks
  - PGAs (Priority Gas Auction) eliminated completely as gas fee no longer determines position in a block
- Leads to decreased competition for block space
  - More genuine transactions get included
  - Eliminates artificial gas price inflation

Discussion

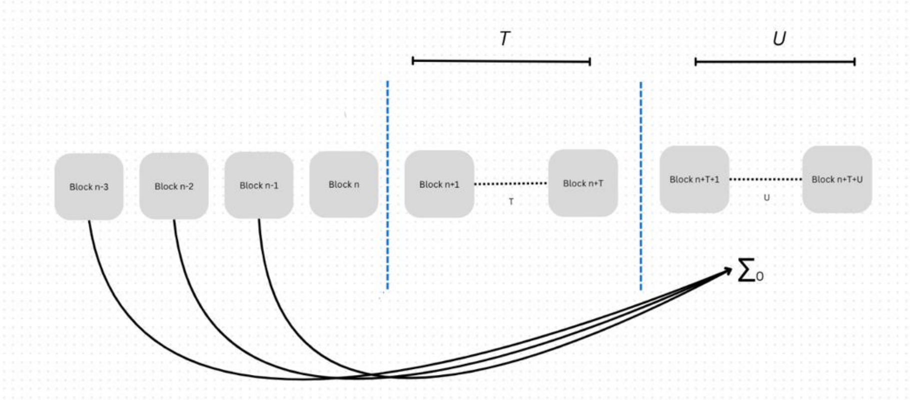
Jostling and Goodput:

- Eliminates sandwich attacks
  - PGAs (Priority Gas Auction) eliminated completely as gas fee no longer determines position in a block
- Leads to decreased competition for block space
  - More genuine transactions get included
  - Eliminates artificial gas price inflation
- Negligible additional overhead (only on-chain space for seeds)
  - Goodput stays high

Discussion

Delay:

- Requires U number of blocks for finalization
  - Exact parameter can be tweaked, at least one block

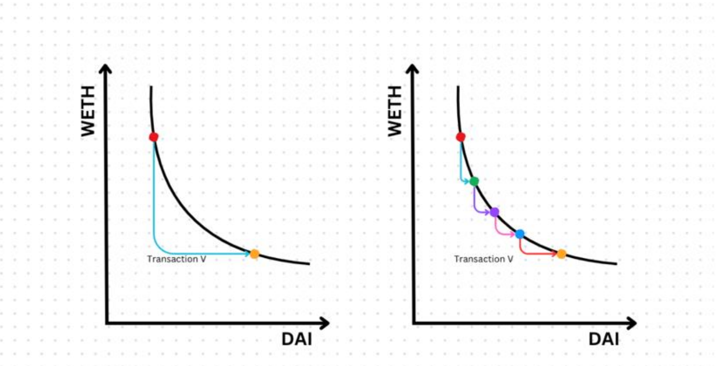


- On Ethereum, can be swapped for RANDAO

Discussion

Cost:

- Introduces no extra cost to the blockchain
- For regular swaps (on CFMM based DEXes), introduces addition (expected) slippage



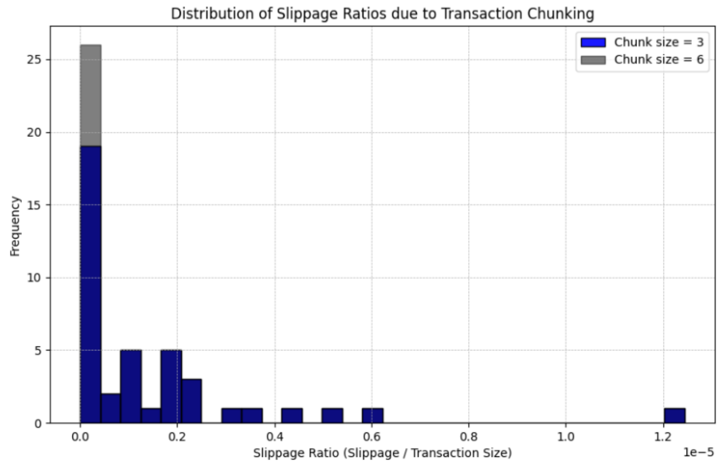
Movement on CFMM curve with a single transaction

Movement on CFMM curve with the same transaction chunked

Discussion

Cost:

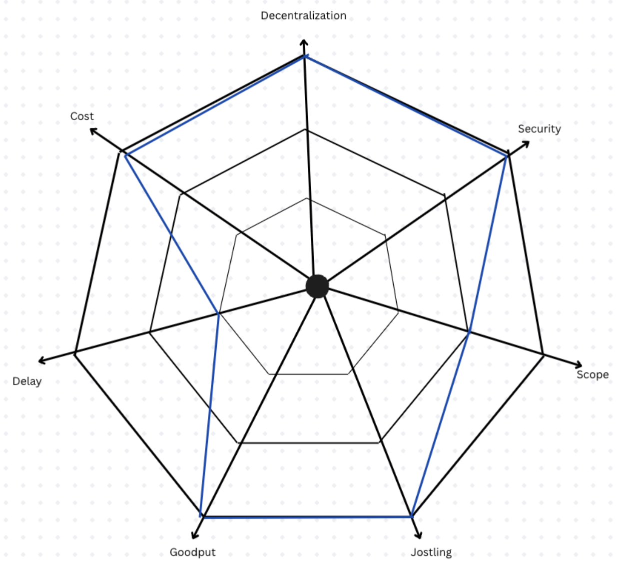
- Slippage higher for transactions of bigger size – constant depending on the chunk size



- Negligible for real-world scenarios

Conclusion

- Can be implemented on existing blockchain without introducing consensus changes
- (Fatal) Front-running becomes a gamble, however not eliminated entirely
- Introduces a minimum of one block of delay
  - May or may not be acceptable depending on blockchain requirements



Future work

- Ethereum already has a randomness generator built in, RANDAO
  - Not perfect, however control over RANDAO is limited to one bit – sufficient entropy for hash functions
  - Protocol can be upgraded to rely on only one leader

- Bigger dataset for evaluation, different AMMs

Questions

Thank you!