Preventing Transaction Reording Manipulations in DeFi Summary

Jack Willis

Introduction

This presentation is a synapsis of Lioba Heimbach and Roger Wattenhofer's paper *SoK: Preventing Transaction Reording Manipulations in Decentralized Finance*

In this paper current methods on preventing reording attacks in defi are analyzed in several different areas on a scale from 1 – 3 (higher is better)

Introduction: What's the problem

- There are currently no schemas that fully meet all the demands of the blockchain ecosystem to prevent reording attacks
  - These attacks occur when an adversary sends their transaction between the invocation and execution of a valid transaction

- This is problematic since smart contracts that power defi are transaction order dependent, the outcome of transactions depend on the order in which they are executed
  - DEXes and lending protocols are primary targets for these attacks

Introduction: Why perform these attacks

- Since miners decide which transactions to include in the next block this gives rise to *blockchain extractable value* (BEV)
  - BEV is a measure of the profit one can obtain by manipulating the order of transactions included in a block

- BEV acts as an invisible tax on transactions leading to increased gas costs for the Ethereum network
  - They can also cause *price gas auctions* (PGA) where attackers compete against each other for BEV by bidding higher gas prices for block inclusion

Introduction: Measures of current solutions

- All BEV mitigation algorithms succeed in some of the following measures but do so at the expense of others
  - Decentralization: the impact a method has on decentralization
  - Scope: the number of different contexts a method is useful in
  - Jostling: the impact a method has on the competition between traders for block inclusion
  - Goodput: the impact a method has on the number of valid transactions processed per unit time
  - Delay: the impact a method has on the delay between transaction submission and execution
  - Cost: how much additional cost a method burdens transactions with
  - Security: a measure of how effective a method is at preventing specific attacks

Reording Attacks

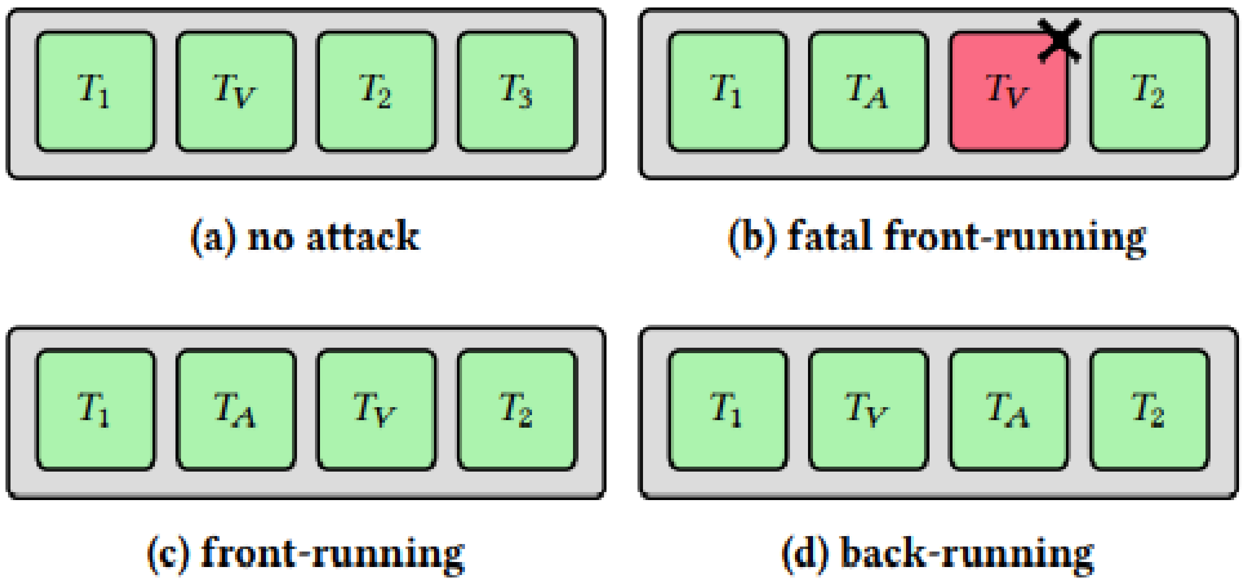
Reording Attacks: How they happen

- When a user performs a transaction on the Ethereum network it enters the public mempool which is a waiting area for transactions to be executed
  - Miners select transactions from here to include in the next block, usually based on how much gas a transaction is willing to pay

- An attacker observing the mempool can see incoming transactions and then profit by inserting their transaction and changing the transaction order
  - The attacker can set a higher gas price, be the miner themselves or bribe a miner to make this happen

Reording Attacks: Types of attacks

- There are *three* main forms of reording attacks:
  - Fatal front-running: when an attacker places their transaction in front of a victims, causing their transaction to execute first and the victim's one to fail
  - Front-running: same as fatal front-running except the victim's transaction does eventually execute
  - Back-running: when an attacker places their transaction behind a victims, causing their transaction to execute after a victim's transaction executes



Attacks in Swapping & Lending

Automated Market Maker Attacks:
Vulnerability in swapping

- AMMs are a type of DEX that automatically facilitate trades based on their liquidity pools
  - CFMMs (*constant function market makers*) are a subclass of AMMs whose exchange rates are algorithmically determined, ensuring that the product of the number of tokens in a pool remains constant
- Since CFMM transactions must be included in a block, an attacker can perform a reording attack such that the liquidity pool of a DEX is shifted in their favor
  - Front-running and back-running attacks are performed sequentially on DEXes so the attacker can make larger profits, this is known as a sandwich attack
- Non-malicious transactions may be in front of yours awaiting execution, because of this traders set a slippage tolerance which is the maximum price shift a trader will allow in the liquidity pool before their transaction is executed (these tolerances mitigate profits from attacks but don’t remove them entirely)

Automated Market Maker Attacks: Example of an attack

This example assumes very high slippage tolerance

- An attacker observes a transaction in the mempool and determines it's advantageous to attack
- The pools reserves are currently 200 aTokens and 50 bTokens, the victim wants to trade 50 bTokens which would yield 100 aTokens
- The attacker *front-runs* buying 68 aTokens for 26 bTokens
- The victim now only receives 52 aTokens for their 50 bTokens instead of 100 aTokens
- Lastly, the attacker *back-runs* selling their 68 aTokens yielding 58 bTokens producing a 32 bTokens profit

Lending Protocol Attacks: Vulnerability in lending

- Any user can be a lender in a lending protocol, they simply provide crypto to the protocols smart contract and receive interest from borrowers
- Once the value of a loan’s collateral drops below a threshold, determined by a price oracle, the protocol liquidates the loan
- If a liquidator finds a liquidated loan profitable, they can send a transaction to the protocols smart contract to claim the collateral, this is a process an attacker can exploit

Lending Protocol Attacks: Examples of attacks

Attack #1:

- An attacker observes an upcoming oracle update that makes a loan available for liquidation
- The attacker *back-runs* the oracle update, ensuring they're the one selected for liquidation

Attack #2:

- An attacker observes a liquidator's transaction in the mempool to claim a loan that's ready for liquidation
- The attack *fatally front-runs* stealing the liquidated collateral

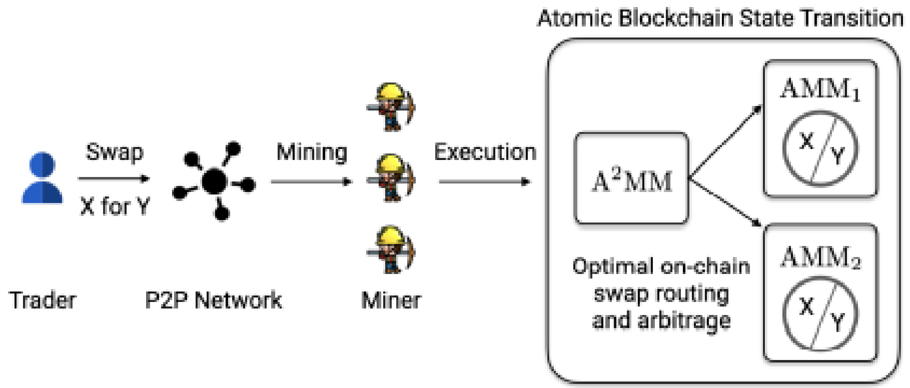
BEV Mitigation Techniques

Optimized Trade Execution: The method

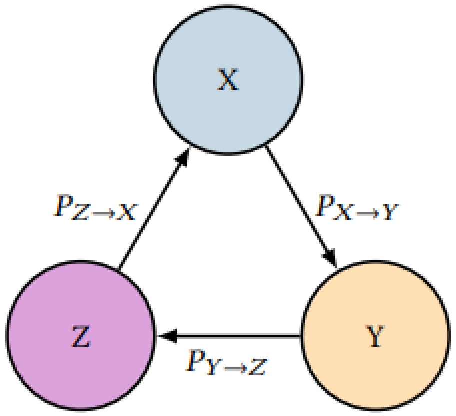
- Optimized Trade Execution are methods that perform application-specific optimizations to mitigate reording attacks
- Examples of Optimized Trade Execution:
  - A scheme known as A^2MM automatically checks if a CPMM user's transaction would create a market imbalance for an attacker is exploit
    - For example, a relatively large trade may create a market imbalance causing a cyclic arbitrage opportunity
  - If an imbalance is caused it automatically collects the arbitrage trade opportunity, leaving no BEV behind
  - Additional schemes of splitting a large transaction into smaller ones and an algorithm that sets transaction slippage could be added to A^2MM mitigating sandwich attacks and transaction failures from natural price changes

Optimized Trade Execution: The method cont.

- A^2MM peers with two AMMs using both liquidity pools. When A^2MM receives a swap transaction, aToken for bToken, it optimizes routing and arbitrage profits among the two AMMs, minimizing any future arbitrage that would attract attackers
- The simplest arbitrage is when a trade is conducted on AMM #1, 50 aToken for 100 bToken, and then another reverse trade on AMM #2, 100 bToken for 75 aToken. If successful, the difference in the AMMs liquidity pools will net a profit

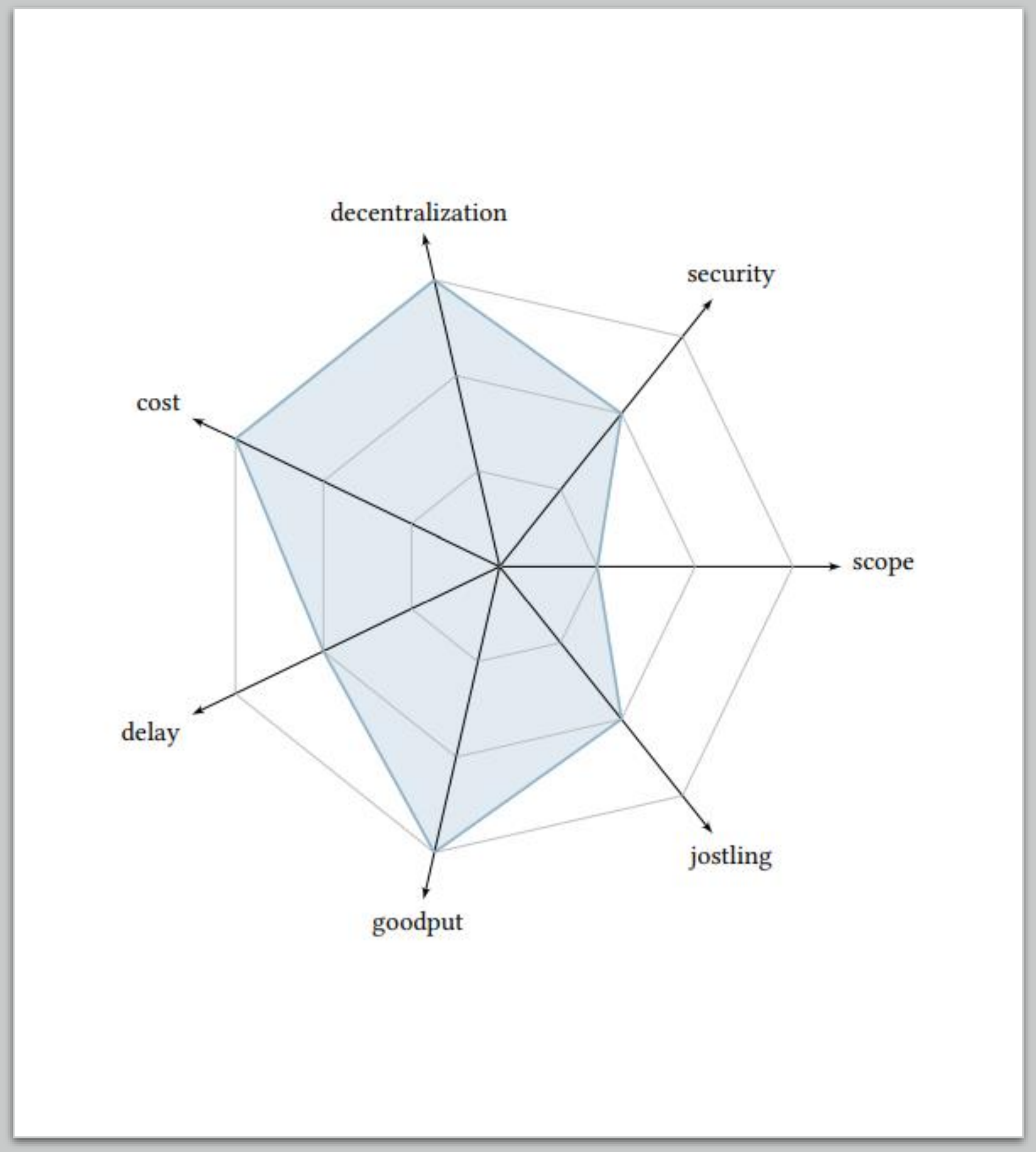


Cyclic Arbitrage:
- Trade 50 aToken for 100 bToken on AMM #1
- Trade 100 bToken for 75 cToken on AMM #2
- Trade 75 cToken on AMM #3 for 100 aToken



Optimized Trade Execution: Outcomes

Decentralization: doesn't impact decentralization (3)
Scope: limited to specific attacks on specific applications (1)
Jostling: could yield a potential increase in competition between traders (2)
Goodput: doesn't affect goodput (3)
Delay: may cause a slight increase in transaction delay since there is time associated with its application to a transaction (2)
Cost: doesn't increase transaction cost (3)
Security: this approach has loopholes stemming from the blockchain's unknown state at execution time (2)



Optimized Trade Execution: Conclusions

- These types of methods are well-suited to act as temporary solutions for users to protect themselves against certain attacks, but mainly fail in the area of scope since their use is limited to specific contexts

Professional Market Makers: The method

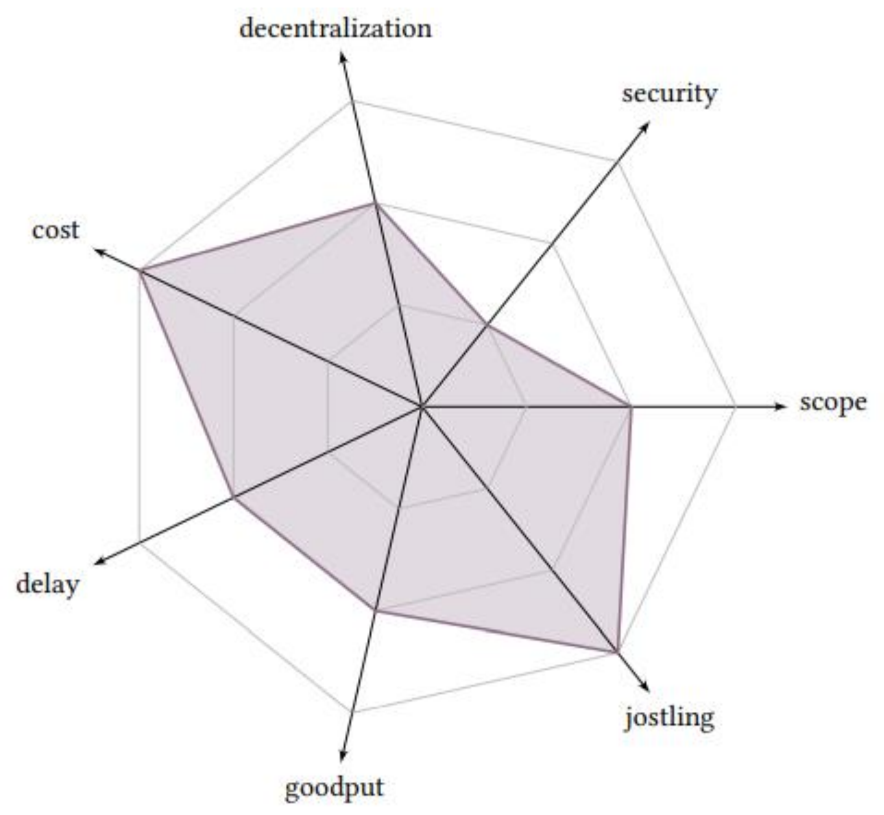
- This technique takes a step further than optimization and completely redesigns DEXes avoiding AMMs entirely
- Example of Professional Market Makers:
  - In the FairMM protocol the buyer sends the trade request off-chain by proposing an exchange rate
  - Once a seller replies, the transaction order is locked and can no longer be tampered with
  - When the price is agreed upon by the market maker and buyer, the transaction executes

Professional Market Makers: The method cont.

- More specifically, in FairMM, a buyer creates a smart contract and locks their 50 aToken in the contract
- Then the buyer sends a request to an individual seller telling them they want to by some bToken
- The exchange rate, let's say for 1 bToken the seller wants 2 aTokens, is decided off-chain, locking the transaction order
- If the buyer agrees with the rate, they send a certificate off-chain to the seller allowing the exchange of 25 bTokens for 50 aTokens
- With this certificate the seller can withdraw the 50 aTokens from the smart contract

Professional Market Makers: Outcomes

Decentralization: impacted by having professional market makers as middlemen (2)
Scope: limited to transaction reording on DEXes specifically (2)
Jostling: doesn't affect competition (3)
Goodput: additional transactions cause a decrease in goodput slightly (2)
Delay: small delay increase because of off-chain communication (2)
Cost: doesn't increase transaction cost (3)
Security: professional market makers can perform reording attacks themselves (1)

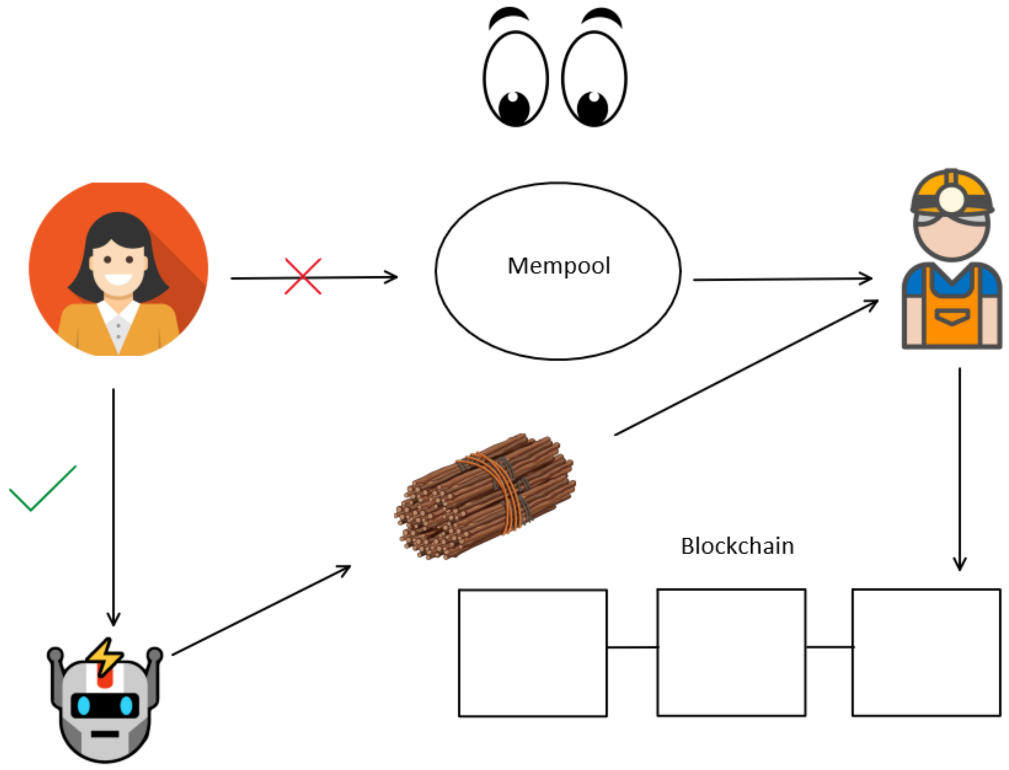


Professional Market Makers: Conclusions

- This method might be difficult to implement since a maker cannot quote a price without knowing the trade size. Additionally, the centralization involved can be seen as a direct violation of blockchain principles

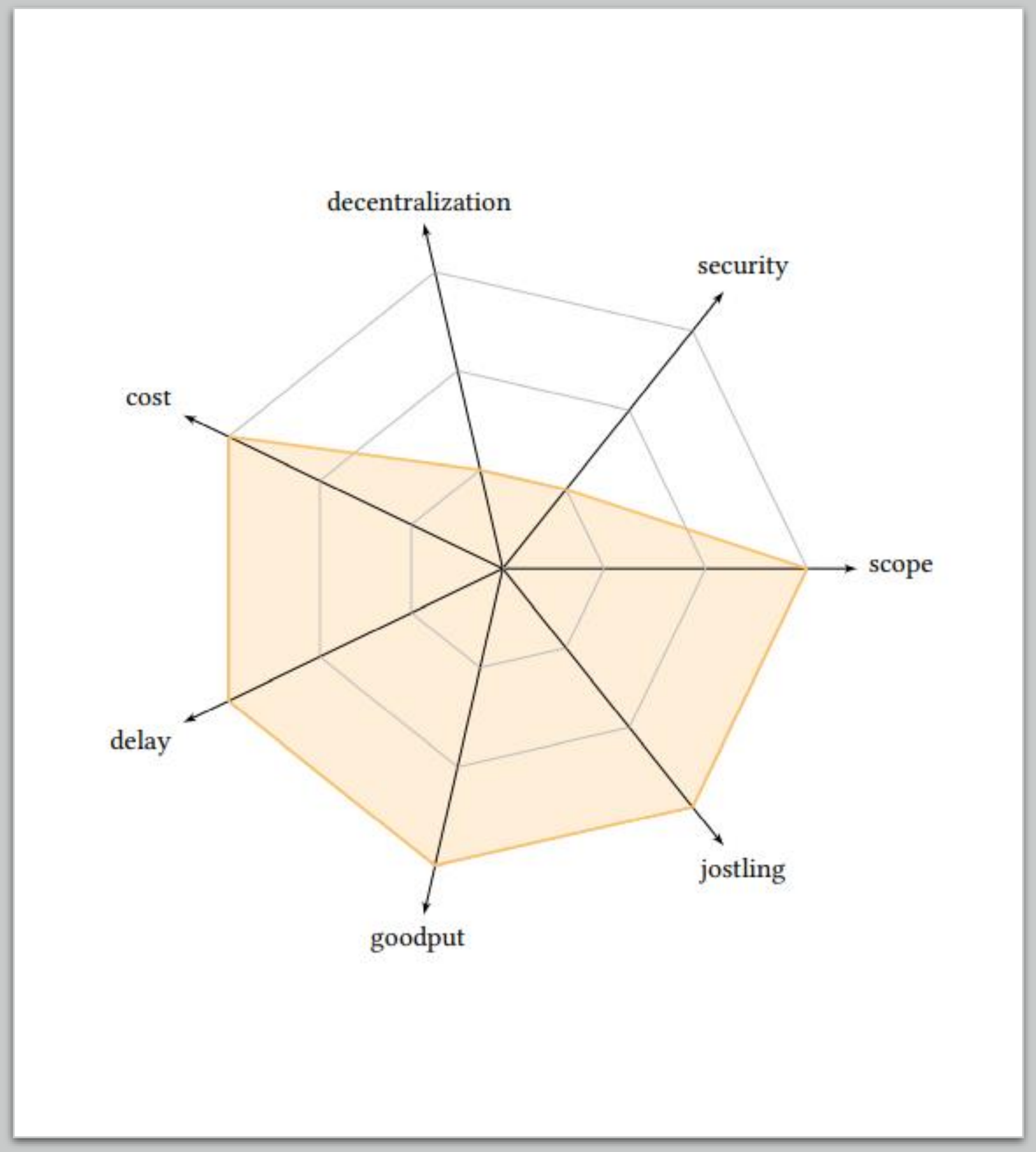
Trusted Third Party Ordering: The method

- This method doesn’t broadcast transactions on the Ethereum network, instead it uses trusted third-party ordering
- Example of Trusted Third-Party Ordering:
  - Users send their transaction to an ordering service like flashbots, Eden, or OpenMEV, where transactions are then bundled in a specific order
  - The bundle is sent directly to miners for block inclusion
  - Transactions bundled by the third-party do not enter the public mempool before execution so cannot be front-run if the third-party is honest



Trusted Third Party Ordering: Outcomes

Decentralization: drastically impacted since a trusted third party is required (1)
Scope: applicable to all types of attacks and platforms discussed (3)
Jostling: doesn't affect competition (3)
Goodput: doesn't affect goodput (3)
Delay: doesn't increase the time needed for a transaction to execute (3)
Cost: doesn't increase transaction cost (3)
Security: very effective at preventing BEV, if the third party can be trusted (1)



Trusted Third Party Ordering: Conclusions

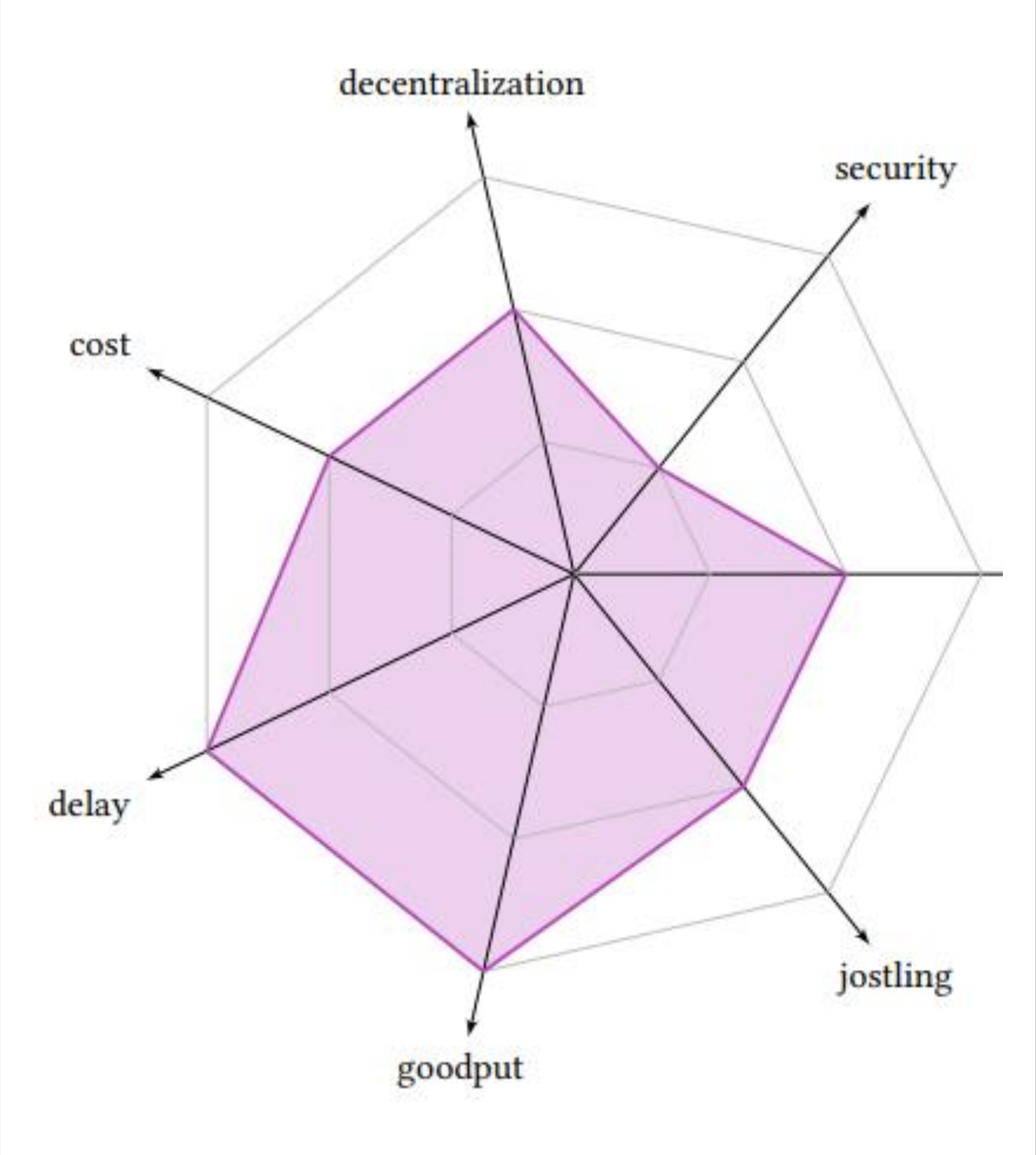
- This method relies on centralized ordering giving it asymmetric performance across the seven measures. This provides the method all the benefits of a centralized entity but contrasts to blockchain principles

Algorithmic Committee Ordering: The method

- This scheme relies on a committee to determine fair ordering of transactions through consensus
- Example of Algorithmic Committee Ordering:
  - A transaction is received by the committee members who then vote on fair ordering via consensus
  - The Condorcet Paradox shows that fairness cannot be completely achieved, but consensus protocols like Hashgraph and Byzantine Ordered Consensus can be utilized to increase fairness accuracy

Algorithmic Committee Ordering: Outcomes

Decentralization: while not completely centralized, this method still relies on committee members, a third party, to reason about ordering (2)
Scope: doesn't mitigate all reordering attacks since they are still possible even with a very honest committee, additionally attacks on the committee are also possible (2)
Jostling: moderate amounts of competition are expected since fatal front-running attacks are potentially possible (2)
Goodput: doesn't affect goodput (3)
Delay: doesn't increase the time needed for a transaction to execute (3)
Cost: slight increase of cost since an incentive for committee members is expected (2)
Security: security is very low since all attacks could potentially be performed on the committee (1)



Algorithmic Committee Ordering: Conclusions

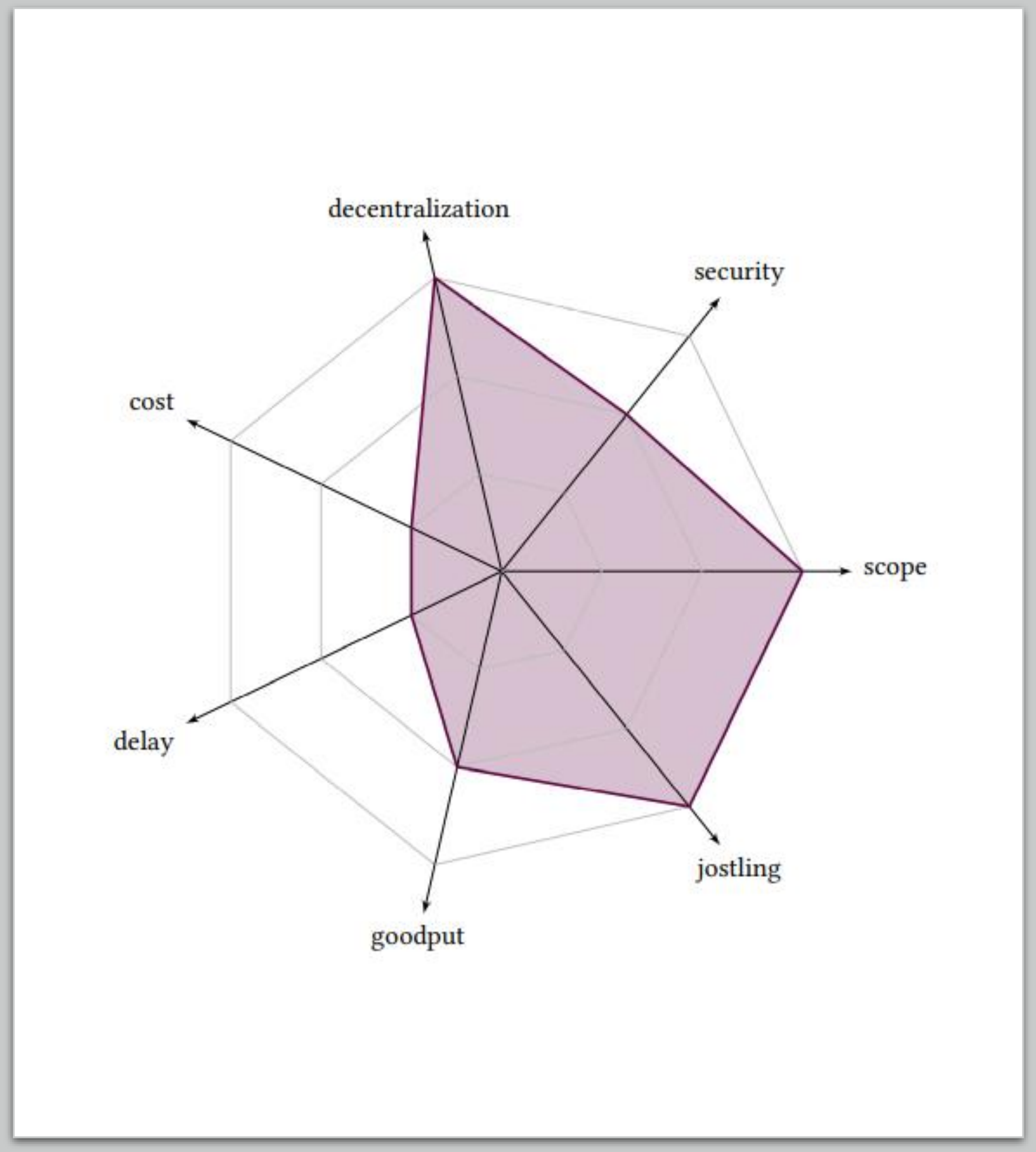
- This method presents a middle ground between trusted third parties and decentralized ordering. However, the possibility of latency related attacks and the trust of the committee limit this methods potential

On-Chain Commit & Reveal: The method

- This method has a committee order transactions on-chain
- Example of On-Chain Commit & Reveal:
  - First users commit their transactions, then the user or the blockchain automatically reveals the transaction in a later block
    - The commit is recorded on-chain and the execution order is determined later

On-Chain Commit & Reveal: Outcomes

Decentralization: doesn't impact decentralization (3)
Scope: protects against all transaction reordering attacks except if there exists and adversarial committee member (3)
Jostling: minimal competition expected (3)
Goodput: due to price movements between commit and reveal increased transaction failure is expected (2)
Delay: significant delay expected since a few blocks must be mined before a committed transactions is revealed (1)
Cost: a DEX using this method cannot reflect prices accurately and because of the delay of execution costs could be significant and cyclic arbitrage opportunities are expected (1)
Security: Small loopholes exist such as an attacker not revealing its transaction (2)



On-Chain Commit & Reveal: Conclusions

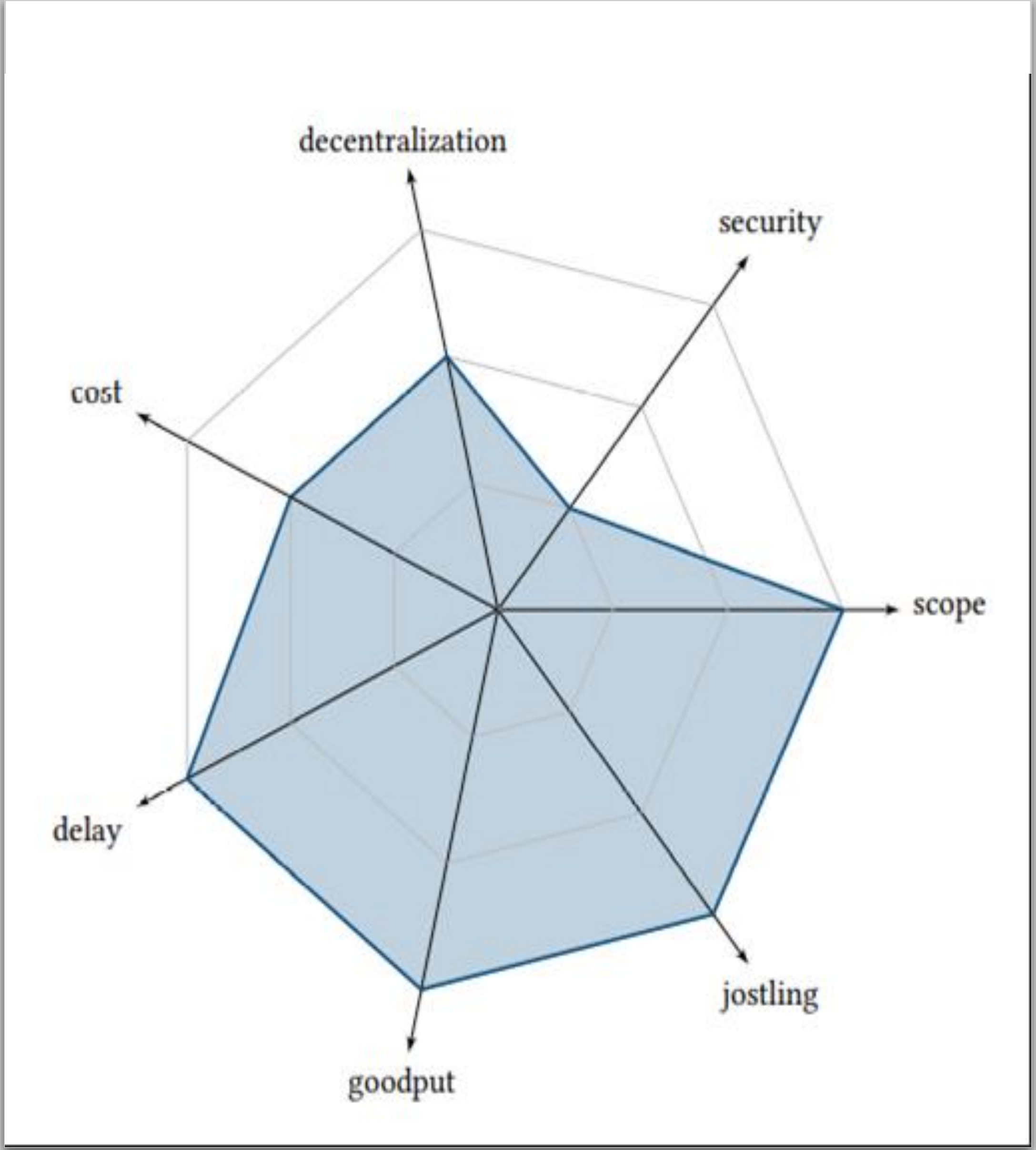
- While this method could be useful outside of DEXes and lending protocols, it is not seen as being able to meet the delay and cost requirements imposed by DEXes

Off-Chain Commit & Reveal: The method

- Like on-chain commit and reveal except this process is done off-chain
- Example of Off-Chain Commit & Reveal:
  - In the first-round traders commit to their transactions off-chain by a committee
  - In the second round the transaction ordering is revealed

Off-Chain Commit & Reveal: Outcomes

Decentralization: because this method is conducted off-chain decentralization is decreased (2)
Scope: protects against all transaction reordering attacks except if the attacker places their transaction first are tackled (3)
Jostling: jostling is reduced since transaction contents are not made public until the orders are fixed (3)
Goodput: doesn't affect goodput (3)
Delay: the committee is not expected to take any longer to verify ordering (3)
Cost: cost is increased since financial incentives for committee members is expected (2)
Security: the lack of decentralization and the committees' ability to perform reordering attacks lead to this approach having poor security (1)



Off-Chain Commit & Reveal: Conclusions

- Except for decentralization and goodput, off-chain commit & reveal appears to generally combine the benefits of algorithmic committee ordering and on-chain commit & reveal. The approach's weaknesses, decentralization and security, are most at odds with the blockchain's fundamental principles

Conclusion

Conclusion: Analysis of mitigation techniques

- We currently see the greatest promise in off-chain commit & reveal to tackle BEV with a general scope, but there are concerns about security
- Optimized trade execution may act well as a temporary fix, lacking mostly in scope
- Despite decentralization and security, third party ordering covers all reording attacks and has excellent performance in goodput, delay, jostling and cost
- While algorithmic committee ordering, off-chain commit & reveal, and professional market makers maintain some decentralization this doesn’t translate to security
    - The former of the three rely on committees which can easily perform manipulations themselves
    - Professional market makers excel in cost and jostling but are otherwise worse then off-chain commit & reveal

Conclusion: Ending remarks

- Mitigating transaction reording manipulations successfully and efficiently on blockchains remains challenging
- Currently, there exists no scheme that meets all the requirements needed for a fully decentralized blockchain
- Hope for a method remains since widespread adoption of trusted third-party ordering is presently underway, conflicting with the foundational ideas of blockchain