Report on Privacy Tech for Building a Dark Pool in DeFi

Misc

- Report from the breakout room on 13 July duringDarkpool.design
- Participants:
- Roshan
- Alan Cisco
- Takeshi
- Eomji Park
- Filip Rezabek
- Other super smart and awesome people

- Roshan
- Alan Cisco
- Takeshi
- Eomji Park
- Filip Rezabek
- Other super smart and awesome people

- Moderator: Lily J
- Authors of the writeup
- Filip Rezabek, @rezabfil
- Lily J, @Lilyjjo

- Filip Rezabek, @rezabfil
- Lily J, @Lilyjjo

In this discussion group, we explored the tradeoffs of using different privacy-enabled technologies (TEE, ZKP, FHE, and MPC) in designs for dark pools. This write-up summarizes the topics we covered, including the different types of trading systems that can be put into a dark pool, the characteristics of the dark pool we care about, an example of the architecture of a dark pool, and the tradeoffs of using different privacy tech for implementations.

1. Introduction

- 1.1 Current Activities Regarding Privacy Enabling Technologies (PETs)
- PETs are becoming a trending topic:[1], [2]
- PETs themselves are tools offering certain properties and tradeoffs
- Our focus: apply PETs to a specific use case, namely Dark Pools and assess the tradeoffs in that setting

- PETs are becoming a trending topic:[1], [2]
- PETs themselves are tools offering certain properties and tradeoffs
- Our focus: apply PETs to a specific use case, namely Dark Pools and assess the tradeoffs in that setting

- 1.2 What is a Dark Pool?
- A dark pool is a private, alternative trading system where investors can trade financial securities without public exposure until after the trade is executed. They are useful for executing large trades where knowledge of the trade would lead to worse price execution. In traditional finance, banks will run their own dark pools for larger institutional clients.

- Other discussion groups in this research day explored questions of best trade types for dark pools, whether there is an equilibrium between lit and dark pool liquidity depth, and how to perform price discovery if all pools are dark, among other topics. This group explored how different PETs would affect the resulting dark pool implementation.
- A dark pool is a private, alternative trading system where investors can trade financial securities without public exposure until after the trade is executed. They are useful for executing large trades where knowledge of the trade would lead to worse price execution. In traditional finance, banks will run their own dark pools for larger institutional clients.

- Other discussion groups in this research day explored questions of best trade types for dark pools, whether there is an equilibrium between lit and dark pool liquidity depth, and how to perform price discovery if all pools are dark, among other topics. This group explored how different PETs would affect the resulting dark pool implementation.

- 1.3 Why do we Need Dark Pools in DeFi?
- DeFi suffers today due to most trade information being public. This leads to front-running attacks, sandwich attacks, and a lack of ability to run private trading strategies. DeFi is also known for being high latency, with trade updates as fast as a block time.
- Dark pools offer potential solutions to these pain points, potentially enabling pre-trade privacy, full trade privacy, private trading strategies, and lower-latency solutions.

- DeFi suffers today due to most trade information being public. This leads to front-running attacks, sandwich attacks, and a lack of ability to run private trading strategies. DeFi is also known for being high latency, with trade updates as fast as a block time.
- Dark pools offer potential solutions to these pain points, potentially enabling pre-trade privacy, full trade privacy, private trading strategies, and lower-latency solutions.

2. Order Processing Styles in Dark Pools

What are the different types of trading systems that we could put into a dark pool?

- 2.1 Overview of Existing Order Processing Styles
- Order Book:
- Description and traditional use: An order book is a real-time list of buy and sell orders for a specific asset. It is commonly used in traditional and decentralized exchanges for continuous matching of trades based on price-time priority.
- Pros: Provides transparency and real-time liquidity; a clear view of market depth.
- Cons: Susceptible to front-running and market impact from large orders.

- Description and traditional use: An order book is a real-time list of buy and sell orders for a specific asset. It is commonly used in traditional and decentralized exchanges for continuous matching of trades based on price-time priority.
- Pros: Provides transparency and real-time liquidity; a clear view of market depth.
- Cons: Susceptible to front-running and market impact from large orders.

- Batching:
- Description and traditional use: Orders are collected into batches and processed together at specific intervals. This method is used to reduce the visibility of individual orders and protect against front-running.
- Pros: Reduces front-running risk; offers greater privacy and anonymity.
- Cons: Introduces latency as orders are not processed in real-time.

- Description and traditional use: Orders are collected into batches and processed together at specific intervals. This method is used to reduce the visibility of individual orders and protect against front-running.
- Pros: Reduces front-running risk; offers greater privacy and anonymity.
- Cons: Introduces latency as orders are not processed in real-time.

- Automated Market Maker (AMM):
- Description and traditional use: AMMs use mathematical formulas to price assets and match orders from liquidity pools

instead of an order book. Common in decentralized exchanges like Uniswap.

- Pros: Simplifies trading; provides continuous liquidity without needing an order book.
- Cons: Liquidity providers are vulnerable to impermanent loss and LVR. Traders are vulnerable to sandwich attacks and price slippage for large trades.

- Description and traditional use: AMMs use mathematical formulas to price assets and match orders from liquidity pools instead of an order book. Common in decentralized exchanges like Uniswap.
- Pros: Simplifies trading; provides continuous liquidity without needing an order book.
- Cons: Liquidity providers are vulnerable to impermanent loss and LVR. Traders are vulnerable to sandwich attacks and price slippage for large trades.

- Request for Quote (RFQ):
- Description and traditional use: Traders request quotes from market makers or liquidity providers and choose to accept or reject these quotes. Common in over-the-counter (OTC) trading. In decentralized systems, liquidity providers often compete to fill a user’s parameterized trade.
- Pros: Enables efficient price discovery; provides competitive and customized quotes.
- Cons: Relies on market makers for liquidity; slower process due to the back-and-forth of quote requests. Off-chain systems can lack visibility and can be restricted to certain liquidity providers.

- Description and traditional use: Traders request quotes from market makers or liquidity providers and choose to accept or reject these quotes. Common in over-the-counter (OTC) trading. In decentralized systems, liquidity providers often compete to fill a user’s parameterized trade.
- Pros: Enables efficient price discovery; provides competitive and customized quotes.
- Cons: Relies on market makers for liquidity; slower process due to the back-and-forth of quote requests. Off-chain systems can lack visibility and can be restricted to certain liquidity providers.

- Order Book:
- Description and traditional use: An order book is a real-time list of buy and sell orders for a specific asset. It is commonly used in traditional and decentralized exchanges for continuous matching of trades based on price-time priority.
- Pros: Provides transparency and real-time liquidity; a clear view of market depth.
- Cons: Susceptible to front-running and market impact from large orders.

- Description and traditional use: An order book is a real-time list of buy and sell orders for a specific asset. It is commonly used in traditional and decentralized exchanges for continuous matching of trades based on price-time priority.
- Pros: Provides transparency and real-time liquidity; a clear view of market depth.
- Cons: Susceptible to front-running and market impact from large orders.

- Batching:
- Description and traditional use: Orders are collected into batches and processed together at specific intervals. This method is used to reduce the visibility of individual orders and protect against front-running.
- Pros: Reduces front-running risk; offers greater privacy and anonymity.
- Cons: Introduces latency as orders are not processed in real-time.

- Description and traditional use: Orders are collected into batches and processed together at specific intervals. This method is used to reduce the visibility of individual orders and protect against front-running.
- Pros: Reduces front-running risk; offers greater privacy and anonymity.
- Cons: Introduces latency as orders are not processed in real-time.

- Automated Market Maker (AMM):
- Description and traditional use: AMMs use mathematical formulas to price assets and match orders from liquidity pools instead of an order book. Common in decentralized exchanges like Uniswap.

- Pros: Simplifies trading; provides continuous liquidity without needing an order book.
- Cons: Liquidity providers are vulnerable to impermanent loss and LVR. Traders are vulnerable to sandwich attacks and price slippage for large trades.
- Description and traditional use: AMMs use mathematical formulas to price assets and match orders from liquidity pools instead of an order book. Common in decentralized exchanges like Uniswap.

- Pros: Simplifies trading; provides continuous liquidity without needing an order book.
- Cons: Liquidity providers are vulnerable to impermanent loss and LVR. Traders are vulnerable to sandwich attacks and price slippage for large trades.

- Request for Quote (RFQ):
- Description and traditional use: Traders request quotes from market makers or liquidity providers and choose to accept or reject these quotes. Common in over-the-counter (OTC) trading. In decentralized systems, liquidity providers often compete to fill a user’s parameterized trade.
- Pros: Enables efficient price discovery; provides competitive and customized quotes.
- Cons: Relies on market makers for liquidity; slower process due to the back-and-forth of quote requests. Off-chain systems can lack visibility and can be restricted to certain liquidity providers.

- Description and traditional use: Traders request quotes from market makers or liquidity providers and choose to accept or reject these quotes. Common in over-the-counter (OTC) trading. In decentralized systems, liquidity providers often compete to fill a user’s parameterized trade.
- Pros: Enables efficient price discovery; provides competitive and customized quotes.
- Cons: Relies on market makers for liquidity; slower process due to the back-and-forth of quote requests. Off-chain systems can lack visibility and can be restricted to certain liquidity providers.

- 2.2 Chosen Method for Discussion: Batching
- Why Batching?
- One of the privacy technologies we wanted to consider included MPC. We were concerned that any other type of method would be too computationally complex for current MPC capabilities to handle, given the need for a trading system to have low latency.

- One of the privacy technologies we wanted to consider included MPC. We were concerned that any other type of method would be too computationally complex for current MPC capabilities to handle, given the need for a trading system to have low latency.
- Why Batching?
- One of the privacy technologies we wanted to consider included MPC. We were concerned that any other type of method would be too computationally complex for current MPC capabilities to handle, given the need for a trading system to have low latency.
- One of the privacy technologies we wanted to consider included MPC. We were concerned that any other type of method would be too computationally complex for current MPC capabilities to handle, given the need for a trading system to have low latency.

3. Evaluation of PETs for Dark Pools

What do we care about when designing a Dark Pool?

- 3.1 Implementation Characteristics
- Scalability:
- How does the system’s performance degrade with increased load?

- How does the system’s performance degrade with increased load?

- Performance:
- What is the latency for a trade? What is the latency for returning information about the trade’s execution and how does that affect the privacy of the overall system?

- What is the latency for a trade? What is the latency for returning information about the trade’s execution and how does that affect the privacy of the overall system?
- Security/trustworthiness:
- What are the trust assumptions and attack vectors of the implementation? What actors need to collude or be dishonest for the system to collapse? What information can be leaked through the allowed user operations?
- What are the trust assumptions and attack vectors of the implementation? What actors need to collude or be dishonest for the system to collapse? What information can be leaked through the allowed user operations?
- User Experience:
- What is the system’s user experience compared to existing systems?
- What is the system’s user experience compared to existing systems?
- Decentralization:
- What entities can affect the functioning of the system? What actors need to be ‘online’ for the system to be accessible?
- What entities can affect the functioning of the system? What actors need to be ‘online’ for the system to be accessible?
- Composability:
- How easy/difficult is it for other decentralized protocols to interact with the system?
- How easy/difficult is it for other decentralized protocols to interact with the system?
- Access control:
- Can a user choose to reveal more information about their trade intent to get better execution, or conversely, hide more information for increased privacy?
- Can a user choose to reveal more information about their trade intent to get better execution, or conversely, hide more information for increased privacy?
- Scalability:
- How does the system’s performance degrade with increased load?
- How does the system’s performance degrade with increased load?
- Performance:
- What is the latency for a trade? What is the latency for returning information about the trade’s execution and how does that affect the privacy of the overall system?
- What is the latency for a trade? What is the latency for returning information about the trade’s execution and how does that affect the privacy of the overall system?
- Security/trustworthiness:
- What are the trust assumptions and attack vectors of the implementation? What actors need to collude or be dishonest for the system to collapse? What information can be leaked through the allowed user operations?
- What are the trust assumptions and attack vectors of the implementation? What actors need to collude or be dishonest for the system to collapse? What information can be leaked through the allowed user operations?
- User Experience:
- What is the system’s user experience compared to existing systems?
- What is the system’s user experience compared to existing systems?
- Decentralization:
- What entities can affect the functioning of the system? What actors need to be ‘online’ for the system to be accessible?
- What entities can affect the functioning of the system? What actors need to be ‘online’ for the system to be

accessible?

- Composability:
- How easy/difficult is it for other decentralized protocols to interact with the system?
- How easy/difficult is it for other decentralized protocols to interact with the system?
- Access control:
- Can a user choose to reveal more information about their trade intent to get better execution, or conversely, hide more information for increased privacy?
- Can a user choose to reveal more information about their trade intent to get better execution, or conversely, hide more information for increased privacy?
- 3.2 Overview Architecture
- We decided to pick an example of architecture to center our discussion around. The architecture consists of a smart contract on a blockchain, where users can lock up liquidity to then be traded on an off-chain dark pool:

[

Dark Pool Architecture

1212×465 18.9 KB

](https://collective.flashbots.net/uploads/default/original/2X/9/93440eaaf6ca29d369ff0c2263427edcc24e0950.png)

- We decided to pick an example of architecture to center our discussion around. The architecture consists of a smart contract on a blockchain, where users can lock up liquidity to then be traded on an off-chain dark pool:

[

Dark Pool Architecture

1212×465 18.9 KB

](https://collective.flashbots.net/uploads/default/original/2X/9/93440eaaf6ca29d369ff0c2263427edcc24e0950.png)

- 3.3 Privacy Enhancing Technologies (PETs)
- We explored using Zero Knowledge Proofs (ZKPs), Fully Homomorphic Encryption (FHE), Multi-party Computation (MPC) and Trusted Execution Environments (TEEs) as options to enable the privacy needed for a Dark Pool implementation. With the reasoning below, we reduced the design space to 1 TEE, 100 TEEs, or MPC.
- 3.3.1 Zero-Knowledge Proofs (ZKPs)
- Used to prove that a computation is correct without revealing the data used during computation.
- Trade-offs: privacy vs. computational overhead.
- Can ZKPs build a dark pool alone?
- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order’s processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order’s processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- 3.3.2 Fully Homomorphic Encryption (FHE)
- Allows computations to be performed on encrypted data, with the results also encrypted.

- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.
- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.
- 3.3.3 Multi-Party Computation (MPC)
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- 3.3.4 Trusted Execution Environment (TEE)
- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.
- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.
- Used to prove that a computation is correct without revealing the data used during computation.
- Trade-offs: privacy vs. computational overhead.
- Can ZKPs build a dark pool alone?
- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order's processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.
- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order's processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling

secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- 3.3.2 Fully Homomorphic Encryption (FHE)
- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.
- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.
- 3.3.3 Multi-Party Computation (MPC)
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- 3.3.4 Trusted Execution Environment (TEE)
- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.
- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.
- We explored using Zero Knowledge Proofs (ZKPs), Fully Homomorphic Encryption (FHE), Multi-party Computation (MPC) and Trusted Execution Environments (TEEs) as options to enable the privacy needed for a Dark Pool implementation. With the reasoning below, we reduced the design space to 1 TEE, 100 TEEs, or MPC.
- 3.3.1 Zero-Knowledge Proofs (ZKPs)
- Used to prove that a computation is correct without revealing the data used during computation.
- Trade-offs: privacy vs. computational overhead.
- Can ZKPs build a dark pool alone?
- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the

actual computation on encrypted data. For example, while they can prove that an order’s processing was valid, they do not provide a mechanism for securely processing the orders.

- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order’s processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- 3.3.2 Fully Homomorphic Encryption (FHE)
- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.

- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.

- 3.3.3 Multi-Party Computation (MPC)
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.

- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.

- 3.3.4 Trusted Execution Environment (TEE)
- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.

- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.

- Used to prove that a computation is correct without revealing the data used during computation.
- Trade-offs: privacy vs. computational overhead.
- Can ZKPs build a dark pool alone?
- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order’s processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- No. ZKPs are used to prove the validity of information without revealing the information itself but cannot handle the actual computation on encrypted data. For example, while they can prove that an order’s processing was valid, they do not provide a mechanism for securely processing the orders.
- They need to be combined with other PETs like FHE, MPC, or TEEs to achieve a comprehensive solution enabling secure computation. These technologies can work together to ensure that all stages of the trading process, from order submission to matching and execution, are secure and private.

- 3.3.2 Fully Homomorphic Encryption (FHE)
- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.

- Allows computations to be performed on encrypted data, with the results also encrypted.
- Trade-offs: strong privacy vs. performance and latency.
- FHE should always be used in conjunction with MPC for key management.

- 3.3.3 Multi-Party Computation (MPC)
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.

- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- Enables multiple parties to jointly compute a function over their inputs while keeping those inputs private.
- Trade-offs: privacy and distributed trust vs. communication overhead.
- Comments:
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.
- We treat FHE and MPC as a combined set for simplicity in our further discussion.
- Similarly, we treat threshold decryption as a special case of MPC.

- 3.3.4 Trusted Execution Environment (TEE)
- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.

- Provides a secure area within a processor where code and data can be executed in isolation from the rest of the system.
- Trade-offs: real-time performance vs. hardware manufacturer trust and many known vulnerabilities. These trade-offs can be modulated by using networks of TEEs instead of a single TEE.
- 3.4 Assessment of Each PET in the Context of Dark Pools
- We chose to compare 1 TEE, 100 TEEs, and MPC/FHE as options to enable privacy in a Dark Pool implementation. Note: 100 TEEs is a random number aiming to provide a feeling of scale/decentralization; 3 or 10 would also be reasonable.
- The communication/agreement among 100 TEEs is not covered in detail, but we assume consensus among the individual nodes as implemented by, e.g., a Secret network.

- We chose to compare 1 TEE, 100 TEEs, and MPC/FHE as options to enable privacy in a Dark Pool implementation. Note: 100 TEEs is a random number aiming to provide a feeling of scale/decentralization; 3 or 10 would also be reasonable.
- The communication/agreement among 100 TEEs is not covered in detail, but we assume consensus among the individual nodes as implemented by, e.g., a Secret network.

1 TEE

100 TEEs

FHE/MPC

Scalability

Highest throughput due to lowest computational overhead.

Lower throughput due to networking costs.

The lowest throughput is due to computational costs associated with MPC. Complex designs could make the system too slow for trading in general.

Performance

Same as above.

Same as above.

Same as above.

Decentralization

Lowest due to a single TEE being a single point of failure. TEE's physical location is important if the system is to be used globally.

Higher decentralization as a single TEE being hacked or broken wouldn't take down the system.

Higher decentralization, but collusion is still possible.

Access Control

TEEs are versatile and can handle complex options. Letting users change their information leakage is doable.

Same as 1 TEE.

Computation is much more expensive, and enabling more complex operations could slow the system down.

Security

Hardware and operator trust assumptions.

Hardware and operator trust assumptions. Better than 1 TEE for the computation output, having multiple TEEs agree on incorrect computing is harder to achieve. Same as 1 TEE for leakage of input batch if TEE is compromised (assuming all TEEs see the same inputs).

Collusion between nodes can leak data.

4. Suggested Advanced Designs for Dark Pools

Below are suggestions on how to use different PETs for Dark Pool designs.

- 4.1 Design 1: Batching with TEEs and ZKPs
- Order Submission:
- Use of ZKPs for order validity.
- Use of ZKPs for order validity.
- Order Matching:
- TEEs for secure processing and matching.
- TEEs for secure processing and matching.
- Execution and Verification:
- ZKPs for on-chain verification of matched orders.
- ZKPs for on-chain verification of matched orders.
- Order Submission:
- Use of ZKPs for order validity.
- Use of ZKPs for order validity.
- Order Matching:
- TEEs for secure processing and matching.
- TEEs for secure processing and matching.
- Execution and Verification:
- ZKPs for on-chain verification of matched orders.
- ZKPs for on-chain verification of matched orders.
- 4.2 Design 2: Batching with FHE/MPC and ZKPs
- Order Submission:
- Fully encrypted orders using FHE. Key management using MPC.
- Fully encrypted orders using FHE. Key management using MPC.
- Order Matching:
- Homomorphic computations for order matching.
- Homomorphic computations for order matching.
- Execution and Verification:
- ZKPs are used to ensure the correctness of the matching process.
- ZKPs are used to ensure the correctness of the matching process.
- Order Submission:
- Fully encrypted orders using FHE. Key management using MPC.
- Fully encrypted orders using FHE. Key management using MPC.
- Order Matching:
- Homomorphic computations for order matching.
- Homomorphic computations for order matching.
- Execution and Verification:

- ZKPs are used to ensure the correctness of the matching process.

- ZKPs are used to ensure the correctness of the matching process.

5. Conclusion & Takeaways

We think that TEEs are versatile and ready for Dark Pools today. In the future, MPC/FHE could offer competing solutions if the technology improves. ZKPs are great for providing an additional layer of validation but can't be used for implementation on their own.

Shoutout to the Flashbots and PropellerHeads teams for creating the space for this design discussion