Abstract:

Decentralized Exchange (DEX) offers transparency and efficiency in trading assets on the blockchain. However, these benefits come with significant privacy and security concerns. This paper explores privacy vulnerabilities in our project DEX implementation, focusing on exposed user data such as order books, user balance and transaction events. When published on a blockchain, malicious actors can exploit these vulnerabilities for frontrunning attacks and undermine user privacy. To address these vulnerabilities, we evaluate and propose three mitigation strategies including stealth addresses, mixers and our custom solution called ‘BatchOrder Protocol’. These strategies aim to enhance privacy and create a more secure decentralized exchange.

Introduction:

Blockchain technology has revolutionized how we store data and transact information and assets over the internet. Its decentralized and immutable nature has allowed for the growing ecosystem of decentralized applications (DApps). With the benefits of blockchain, it also introduced three significant challenges: security, privacy and scalability.

The security of the blockchain often refers to the vulnerabilities in the smart contract, consensus, runtime and P2P Network layers. Scalability refers to the difficulty and challenges of maintaining state validity and data availability. Privacy pertains to the transparency of the blockchain transactions and contract, which can expose sensitive user data.

In the past, there have been multiple instances where researchers are able to gather and analyze transaction graphs of Bitcoin. Some of these includes:

1. In 2012, Dorit Ron and Adi Shamir did a Quantitative Analysis of the Full Bitcoin Transaction Graph and demonstrate how linking techniques can connect Bitcoin addresses, exposing patterns and undermining user privacy. [1]
2. In 2013, researchers applied clustering techniques to link Bitcoin addresses to entities and showed that transactions can be traced, undermining Bitcoin’ privacy. [2]
3. In 2015, researchers linked Bitcoin addresses to real identities through publicly available sources like forums. [3]

For our development project, we designed and implemented a decentralized exchange (DEX), which enables users to trade Ethereum with custom tokens on the blockchain. The goal of our project was to develop an architecture which allows for the management of token balances, order books and transaction execution. While this architecture achieves functional transparency, we realised that it exposes user balances, orders, transactions histories in the blockchain.

In this paper, we will focus on addressing privacy vulnerabilities in our DEX architecture, such as exposed mappings, event logging and order book transparency. We will also propose solutions to protect user anonymity and transaction confidentiality, which includes stealth addresses, mixers, and our custom solution called ‘BatchOrder Protocol’ to enhance the privacy of users without compromising functionality.

Motivation:

In our blockchain lectures, privacy is defined as selective disclosure of information. Privacy is compromised in decentralized exchanges due to the transparency and immutability of blockchain networks. Publicly accessible data, such as wallet balances, transaction history and trading strategies can be exploited by malicious actors. For example, frontrunning attacks allow attackers to profit from visible buy and sell orders by preemptively placing their own trades. In May 2024, a former trader for U.S. pension giant TIAA-CREF was sentenced to 70 months in prison after he engaged in more than a thousand front-running trades between 2016 and the end of 2022, racking up at least US$47 million in illegal profits. [4]

This violates the privacy of users involved in buying and selling tokens. In addition, the ability to potentially trace transaction histories back to a user’s real-world identity undermines user privacy, exposing them to phishing and hacking attempts.

Blockchain Trilemma

Privacy concerns discourage users from participating in decentralised exchange since their financial data and trading strategies are exposed. While scalability is critical, there are established solutions like sharding and rollups that are widely adopted. Our project should implement scalability solutions only when there is high traffic volume. Although security is fundamental, blockchain networks like Ethereum have strong cryptographic guarantees and our project does not introduce additional risks in security beyond the standard practices and implementations such as ERC-20 for token creation.

Privacy concerns became evident nearing the end of the development of our project. The tokenBalanceForAddress mapping exposes user balances, and the getBuyOrderBook and getSellOrderBook functions allow anyone to monitor the order books. Addressing privacy bridges the gap between the blockchain’s transparency and real-world usability, ensuring that users can engage in a secure exchange without compromising sensitive data. This will also ensure broader adoption and long-term viability for our exchange.

Literature survey:

Similar privacy challenges in blockchain and decentralized applications have been studied in academic and industry contexts. Some key works include:

1. Ethereum Whitepaper: Highlights the public nature of Ethereum transactions and the need for privacy-preserving mechanisms in smart contract applications. [5]

2. Frontrunning in DEXs: Demonstrates how adversaries exploit transaction visibility in DEXs for profit through frontrunning attacks. [6]

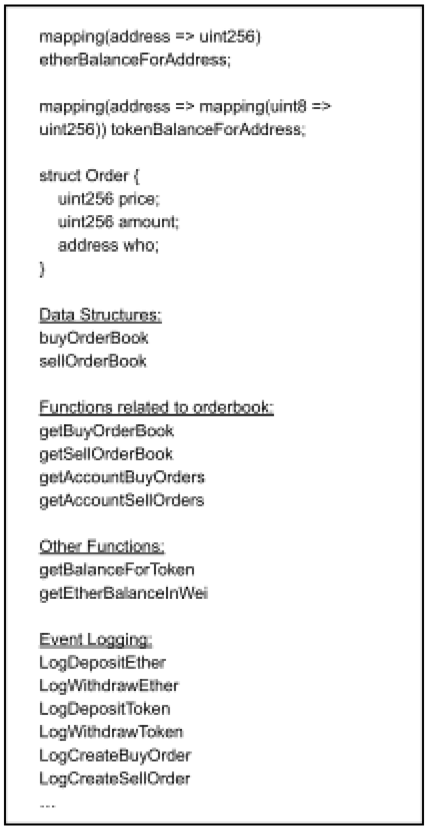
3. Stealth Addresses for Privacy: Introduces a method for generating one-time payment addresses to enhance transaction privacy in Ethereum. [7]

4. Submarine Sends: Proposes commit-reveal schemes for delaying transaction disclosure, thereby mitigating frontrunning and privacy breaches. [8]

5. Understanding Batch Auctions (CowSwap): Demonstrates how Batch Auction works in the CoW Protocol. [9]

Observation and Analysis

Components that are exposed:



```
mapping(address => uint256)
etherBalanceForAddress;

mapping(address => mapping(uint8 =>
uint256)) tokenBalanceForAddress;

struct Order {
   uint256 price;
   uint256 amount;
   address who;
}

Data Structures:
buyOrderBook
sellOrderBook

Functions related to orderbook:
getBuyOrderBook
getSellOrderBook
getAccountBuyOrders
getAccountSellOrders

Other Functions:
getBalanceForToken
getEtherBalanceInWei

Event Logging:
LogDepositEther
LogWithdrawEther
LogDepositToken
LogWithdrawToken
LogCreateBuyOrder
LogCreateSellOrder
...
```

Based on our project, we have TokenManagement.sol and EtherManagement.sol, which define the architecture of the decentralized exchange. In the code segments above, the ethereum and custom tokens for all users, as well as the orders containing the transaction price, amount and receiver, can be read by anyone who queries functions like getBuyOrderBook or getSellOrderBook in the contract or by using tools to read the transactions. In addition, the event loggings should be removed if it is unnecessary.

Malicious actors or competitors could analyze our blockchain data to extract trading patterns, balances or strategies. This information can be used for frontrunning and undermining user privacy.

Issues Identified

Frontrunning attack:

When buy orders are visible to everyone, an attacker can monitor these transactions and place their orders ahead of others to take advantage of price fluctuations or discrepancies. With our order books showing prices and amounts for each order, attackers can query these order books using getBuyOrderBook or getSellOrderBook functions to learn and analyze these in real time. In addition, when we log events using methods like LogCreateBuyOrder or LogCreateSellOrder, these broadcasts can aid attackers in monitoring the system, enabling them to do a front running attack. This form of frontrunning attack is known as insertion attack.

Example scenario that exploits a buy order:

1) A user places a buy order at a higher price to secure tokens.
2) The attacker sees the order and places their buy order with a higher gas fee, ensuring it is mined first.
3) The attacker purchases the tokens at a lower price.
4) After the user's order is executed, the token price rises, and the attacker sells the tokens for profit.

By gaining a control of a larger share of tokens(assuming that our network is still relatively small), we can drive up the price since the demand due to increased demand (or exhaustion of lower-priced sell orders in the order book).

Undermining user privacy:

Our decentralized application includes the names of the functions that can be invoked and the identities of the sender and receiver, who may wish to remain unknown. This undermines user privacy. Tracing of transactions on the chain can potentially reveal and expose sensitive financial information. This allows the public to identify high-value wallets that can be attacked or tie real-world identities to the blockchain addresses. Once high-value wallets or real identities are discovered, attackers can use spear phishing tactics to compromise these wallets. By crafting highly targeted and convincing fraudulent messages, attackers may trick users into revealing private keys by clicking malicious links, or signing transactions that drain their funds. This disincentives users to use the DApp as their identity can be leaked.

Leaked Token Balances

The tokenBalanceForAddress mapping is exposed and the data stored on a blockchain is accessible to anyone to read it. While smart contracts can restrict who can interact with certain functions using access modifiers like private or internal, this alone doesn't fully hide the balances because any on-chain operation such as fulfilBuyOrder and fulfilSellOrder references this mapping and leaves traceable records in the transaction history that can be analyzed to deduce each individual's balance. This violates the privacy of each individual that uses the contract.

How the security issues have been addressed

Possible solutions to fix remaining issues

Stealth addresses

Stealth addresses are unique, one-time addresses generated for each transaction to enhance privacy. A recipient can publish a public key which the sender can use to create a unique address for each payment. Only the recipient can link these addresses to their private key, ensuring that transaction history cannot be publicly correlated with a specific user. This process can be implemented using ERC-5564, the Stealth Address Standard, which defines a method for generating and managing stealth addresses on Ethereum. ERC-5564 allows senders to create unique payment addresses using cryptographic operations specified by the standard, ensuring enhanced privacy.

Functions specified in the standard such as generateStealthAddress takes in a recipient stealthMetaAddress and returns a stealthAddress. A stealthMetaAddress is composed of 2 pair of private keys that are randomly generated and kept by the recipient:

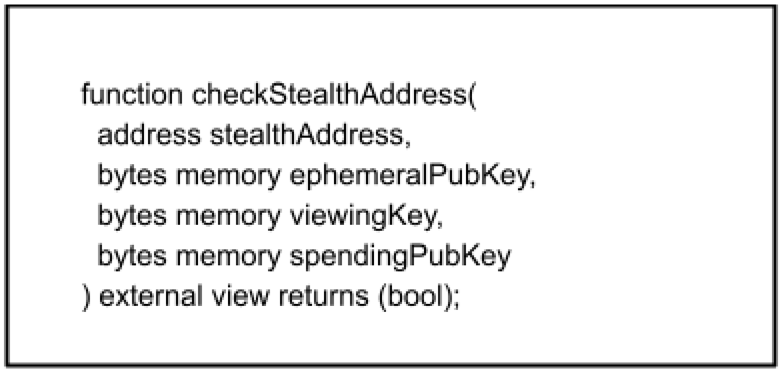
1) Spending Key: allow the recipient to spend funds received at the derived stealth addresses.
2) Viewing Key: allow the recipient to scan the blockchain to detect funds sent to the derived stealth addresses.

The spending and viewing keys are generated during the initial setup of a user's stealth address system. They can be stored in MetaMask or hardware wallets like Ledger or Trezor.

Inside the generateStealthAddress function, the sender would generate a new ephemeral public key from the private key(spending or viewing) for each transaction. This creates the stealth address.

```
function generateStealthAddress(bytes memory stealthMetaAddress)
  external
  view
  returns (address stealthAddress, bytes memory ephemeralPubKey, bytes1 viewTag);
```

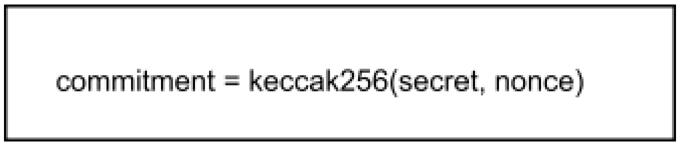
The checkStealthAddress function verify whether a given stealth address is valid and associated with a recipient. Using the ephemeralPubKey, viewingKey and spendingPubKey, we can check whether a stealthAddress is valid.



```
function checkStealthAddress(
  address stealthAddress,
  bytes memory ephemeralPubKey,
  bytes memory viewingKey,
  bytes memory spendingPubKey
) external view returns (bool);
```

Mixing Pool for tokenAddressForBalances
Instead of directly maintaining a mapping of ‘tokenAddressForBalances’, the balances are held in a pooled format. The following is the proposed workflow of the ‘cryptographic pool’:

Deposit:
The user calls a deposit() function and provides a token amount and cryptographic ‘commitment’. The commitment can be a hash of a secret and a user-defined nonce. This commitment is meant to the user details such as his account address and token balance. [12]



```
commitment = keccak256(secret, nonce)
```

The deposit can happen right after both buy and sell orders have been matched, or when the user has initially deposited his tokens to the exchange.

Shuffling:
The tokens and the commitments are stored separately. The tokens are stored in a token pool, while the commitments are stored in memory. The tokens in the token pool are shuffled periodically to ensure that the tokens and commitments are decoupled. Similarly, the

commitments are shuffled periodically to further anonymize the deposits. This pool is managed by the contract responsible for the design of the decentralised exchange.

<u>Withdraw:</u>
When a user wants to withdraw tokens, he shows the commitment by providing the secret and the nonce. If the commitment is valid and unspent, the token pool approves the withdrawal and provides the corresponding tokens, though they may not be the exact tokens the user has deposited or received from a transaction.

BatchOrder Protocol
*This solution is inspired by CowSwap's Batch Auction [11] and adapted according to the requirements of our project.

When a user submits an order (can either be buy or sell order), which is appended to the corresponding buy or sell order book. Since both the buy order book and sell order book were initially initialised as priority queues that sort the elements based on the price in each order, and since we no longer prioritise the price of the orders, the elements in queues are now sorted based on their time of arrival. In other words, FIFO (First In, First Out) queues are initialised for both the buy order book and sell order book.

At a fixed time interval, all the orders from both the buy order book and sell order book are collated into a batch. All the orders within a single batch are executed at a uniform clearing price. The clearing price is set where most trades can be executed. In the diagram below, it is $97. For matched orders, the buy order at $97 matches the sell order at $97, and the buy order at $100 matches the sell order at $95. The buy order at $95 and the sell order at $90 remain unfilled and are moved to the next batch, as their prices do not meet the clearing price.

Buy Orders		
Buy 1 ETH @ $100	Buy 1 ETH @ $97	Buy 1 ETH @ $95

Sell Orders		
Sell 1 ETH @ $95	Sell 1 ETH @ $97	Sell 1 ETH @ $90

Orders are sorted into their respective batches based on their time of arrival, particularly on the time interval they are closest to when a new batch is created.

This solution resolves the issue of front-running attacks because orders are processed simultaneously and not prioritised based on time or position in the order book.

Conclusion:

In this paper, we have focused on the privacy issues of decentralised exchange projects. Along with the issues, we proposed some solutions as well as the steps needed to implement the solution. The proposed solutions ensure that there are mechanisms in-place to protect user's privacy during transactions.

Tackling the issues of user's privacy is an ever on-going effort and while our solutions provide a stronger foundation to our DApp, they may not address every vulnerability. We will have to stay vigilant on news of future attacks or bugs that undermine user privacy and respond to these discovered vulnerabilities promptly to ensure that our decentralized exchange remains safe and private for all users.

[1] Ron, D., & Shamir, A. (n.d.). Quantitative analysis of the full bitcoin transaction graph. https://eprint.iacr.org/2012/584.pdf

[2] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., M. Voelker, G., & Savage, S. (n.d.). A fistful of bitcoins: Characterizing payments among men with no names. https://cseweb.ucsd.edu/~smeiklejohn/files/imc13.pdf

[3] Fleder, M., S. Kester, M., & Pillai, S. (2015, February 5). Bitcoin transaction graph analysis. https://arxiv.org/pdf/1502.01657.pdf

[4] Langton, J. (2024, May 21). *Trader gets jail time for front-running scheme*. Investment Executive. https://www.investmentexecutive.com/news/from-the-regulators/trader-gets-jail-time-for-front-running-scheme/

[5] Bayan , T., & Banach, R. (n.d.). Exploring the privacy concerns in permissionless Blockchain Networks and Potential Solutions https://arxiv.org/pdf/2305.01038

[6] Buterin, V. (2014). *Ethereum whitepaper*. ethereum.org. https://ethereum.org/en/whitepaper/

[7] Philip Daian, P. D., Goldfeder, S., Kell, T., Li, Y., & Zhao, X. (2019, April 10). Flash boys 2.0: Frontrunning, transaction reordering, and Consensus Instability in Decentralized Exchanges. http://arxiv.org/pdf/1904.05234

[8] Wahrst, T., Solomon, M., DiFrancesco, B., & Buterin, V. (2022, August 13). *ERC-5564: Stealth addresses*. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-5564

[9] Shayan , S., & Moosavi, M. (2019, February). (PDF) Sok: Transparent dishonesty: Front-running attacks on Blockchain. https://www.researchgate.net/publication/339890096_SoK_Transparent_Dishonesty_Front-Running_Attacks_on_Blockchain

[10] Understanding batch auctions. CoW DAO. (n.d.). https://cow.fi/learn/understanding-batch-auctions

[11] Kochan, F. (2024, November 26). *Private transactions on Ethereum using stealth addresses (ERC-5564)*. QuickNode Guides. https://www.quicknode.com/guides/ethereum-development/wallets/how-to-use-stealth-addresses-on-ethereum-eip-5564

[12] Kochan, F. (2024a, November 26). *How to use KECCAK256 hash function with solidity*. QuickNode Guides. https://www.quicknode.com/guides/ethereum-development/smart-contracts/how-to-use-keccak256-with-solidity

Security/Privacy/Scalability issues in Title of Your Development Project

Full Name of Student
Student's Matriculation Number
Name of Student's School
Nanyang Technological University
Singapore
Email address of Student

Full Name of Student
Student's Matriculation Number
Name of Student's School
Nanyang Technological University
Singapore
Email address of Student

Full Name of Student
Student's Matriculation Number
Name of Student's School
Nanyang Technological University
Singapore
Email address of Student

***Abstract*—This is a template for the term paper of CE/CZ4153 Blockchain Technology course offered at the School of Computer Science and Engineering, Nanyang Technological University, Singapore. The paper should follow a similar style and format to incorporate all the necessary points of introduction, motivation, literature survey, observations, analysis, and solution to the issue. The paper should be 4 to 6 pages in length, including references. In case extra material is needed for analysis or arguments, the authors may include that as appendices, after the references.**

Keywords—blockchain, Ethereum, smart contracts, tokens, may be some more with respect to your specific issue and project.

I. Introduction

In this section, you should write a few paragraphs on blockchain (brief, can adapt from lectures, in your own words), the issues of security, privacy and scalability in blockchain (brief, can adapt from lectures, in your own words), what your development project topic was, and what you finally developed.

You should also mention exactly which issue out of the three – Security, Privacy, Scalability – you are going to present, and what would be the overall contribution of this paper. Argument as why this issue is the most important is not required here.

II.Motivation and Literature Survey

A. Motivation

In this subsection, you should clearly argue which one of the three issues – Security, Privacy, Scalability – concerns you the most in case of the Decentralized Application you developed. You may refer to the lectures, invited talks, related works, or any other instance of similar development projects to argue this.

B. Literature Survey

Term Paper submitted for CE/CZ4153 Blockchain Technology course, NTU.

In this subsection, you should carefully curate similar works in the area of your interest, with proper citation (see references). This may include similar development projects or decentralized applications that have faced the same issues, lectures, articles, books or papers that talk about the issue in your case, or any other academic material related to your specific case.

III. Observations and Analysis

In this section, start by listing your main observations on the issue you chose in case of your development project. In each case, discuss the major considerations, analyze their impact (and ramifications) on your decentralized application, and compare it with similar cases in the literature, if you found any such case.

A. Issue X in case of Component I

Identify the specific issue X that will affect component Y of your decentralized application (e.g., *re-entry bug in case of the auction contract*). State why you think this issue may occur, what would be the impact on your application, and whether you know of any similar case in the literature where this happened.

B. Issue Y in case of Component I

There may be more than one issue per component. Think carefully to spot all such issues in your development project and write one subsection on each one of them. In case they are connected, do mention that too in this portion of your paper.

C. Issue Z in case of Component II

There may be more than one component with an issue. Think carefully to spot all such issues in your development project and write one subsection on each one of them. In case they are connected, do mention that too in this portion of your paper.

IV. Proposed Solutions

In this section, propose potential solutions to address the issues that you found in your analysis earlier. These solutions may be inspired from the lectures, invited talks, related works, or any other instance of similar development projects.

A. Solution to Issue X

Identify potential solutions to this issue. Clearly mention how you would apply the solution to your development project, and if you have already applied the solution. Applying the solution is of course not mandatory for the development project.

B. Solution to Issue Y

Identify potential solutions to this issue. Clearly mention how you would apply the solution to your development project, and if you have already applied the solution. Applying the solution is of course not mandatory for the development project. In case the solution to issue X already solves Y, mention that.

C. Solution to Issue Z

Identify potential solutions to this issue. Clearly mention how you would apply the solution to your development project, and if you have already applied the solution. Applying the solution is of course not mandatory for the development project. In case there exists no known solution to issue Z, propose a potential solution on your own, and argue why it may work.

V. Conclusion

In this section, you should mention exactly which issue out of the three – Security, Privacy, Scalability – you presented, and what is the overall contribution of this paper. The contribution may be in terms of your observations, analysis or proposed solutions presented for the issues and the components.

You may follow the IEEE paper format for the Tables, Lists, Figures, References, etc. Keep the format uniform in the paper.

References

[1] A. Narayanan, J. Bonneau, E. W. Felten, A. Miller, and S. Goldfeder, “Bitcoin and Cryptocurrency Technologies – A Comprehensive Introduction,” Princeton University Press 2016.

[2] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, “SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies,” IEEE Symp. on Security and Privacy 2015: 104-121.

[3] I. Eyal and E. Gun Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” Financial Cryptography, 2014, pp. 436–454.