Decentralized commit-reveal scheme to defend against front-running attacks on Decentralized EXchanges

***Abstract*—In this paper, we focus on the security issues of decentralized exchanges (DEXs), specifically on front-running attacks. Front-running attacks, which exploit the transparency of blockchain and its transaction processing mechanisms, pose a significant threat by undermining the reliability and efficiency of DEXs and eroding user trust. To solve this problem, this paper proposes a new decentralized commit-reveal scheme that overcomes the limitations of the existing commit-reveal scheme and enhances user experience and the spirit of decentralization. This scheme was developed through Ethereuem's Uniswap, and guarantees safe transactions through the proposed relay router and deputy reveal mechanism and effectively blocks frontrunning attacks. We demonstrate the security and efficiency of the proposed scheme through scenario-based verification and identify a new solution to front-running attacks in the DEX environment.**

Keywords—Decentralized-EXchange, Decentralized Finance, Ethereum, front-running, DEX

I. Introduction

With the evolution of the blockchain technology environment, decentralized exchanges (DEXs) have emerged as a way to provide a platform for trading cryptocurrencies without centralized authority, servers, and administrators. DEX promotes P2P transactions through smart contracts, enhances transparency and security of transactions, and excludes risks such as administrator malicious actions and authority theft that occur in centralized exchanges. Despite these advantages, DEXs face unique challenges, especially in the areas of security and fair trading practices. DEX transactions are disclosed through the blockchain's transaction transparency, and due to this transparency and the difference in transaction processing order depending on the gas price, which is a feature of the blockchain's consensus protocol, it has become a target of attack by automated transaction bots. Among Bot attack methods, front-running attacks undermine the integrity and fairness of the DeFi market, weaken user trust, and pose a serious financial threat [1]. This can make users aware of the risk of manipulated transactions, which can lead to rejection of the DEX environment and hinder the continued development of DeFi.

We collected and analyzed transactions of Uniswap, the largest DEX, through the first quarter of 2023. The average daily attack volume of Maximal Extractable Value (MEV)-Bots, which refers to Bots that take profits in trading history, is approximately $480 million, accounting for 45% of total daily trading volume. MEV-bot performs attacks such as arbitrage and front-running attacks. Front-running is an attack method that utilizes the transparency of blockchain transactions and the block generation mechanism. Front-runners can manipulate market prices to their advantage by intercepting and executing transactions before others based on pending transaction data available in the mempool [2]. This not only results in unfair trading conditions, but also jeopardizes the spirit of the DEX by abusing the transparency that supports the trustworthiness of a DEX with decentralization as its ethos. Traditional defenses against front-running often rely on centralized solutions, which paradoxically undermines the decentralized nature of DEXs. These approaches typically involve third-party services or centralized nodes to implement a commit-reveal scheme, creating potential vulnerabilities and centralization.

We propose a decentralized commit-reveal scheme specifically designed to solve the problem of front-running attacks in DEX. The contribution of this paper maintains the decentralized spirit of DEX, eliminating dependence on centralized entities while effectively mitigating front-running risk. At the same time, the possibility of metadata-based inference attacks is ruled out by ensuring data and metadata confidentiality at the commit stage. In addition, the scheme hardly changes the structure of the current DEX, and meets convenience in terms of user experience, providing unchanging usability on the surface. This ensures that all market participants participate in a fair and transparent trading environment, maintaining the integrity of the DEX, strengthening user trust in these platforms, and mitigating loss points between transactions. Additionally, proposals to limit the behavior of these bots are expected to contribute to the abnormal fee surge [3] in blockchain networks. In Chapter 2, this paper confirms the basic knowledge about blockchain and front-running and various attempts to prevent front-running. The proposed scheme is explained in Chapter 3. To confirm the stability of the proposed scheme, scenario-based verification is performed in Chapter 4, and the conclusion is described in Chapter 5.

II. Front-running attacks on DEX

A. Front-running Attacks

Front-running that occurs in blockchain is based on the data transparency of blockchain and the characteristics of the block generation protocol. In the block generation protocol, transactions sent by users to the blockchain are recorded in the mempool, which is the transaction waiting area. The miner (Ethereum: Validator) first selects transactions that pay high fees (gas) from the mempool and adds them to the block in order to generate maximum profit when creating a block. The mempool, which is transactions that have not yet been added to the block, is public due to data transparency and can be accessed by anyone through node programs such as geth. The

Funding space .

attacker (MEV-Bot) in Fig. 1 checks the remaining transactions and transaction fees in the mempool and transmits a preceding transaction that can be executed with a higher priority than the attack target's transaction. The attacker's transaction is performed before the attack target's transaction, and activities such as DEX swap and liquidity cause market value volatility depending on the time element of information transfer. As a result, the attack target suffers losses due to a value exchange that is different from the intended price. Through this proactive transaction interference, attackers create unfair transaction conditions, cause losses to users and the platform, and steal profits. Front-running attacks that occur in DEX are mainly LP Sandwich attacks and JIT liquidity attacks.

B. *Types of Front-running Attacks*

In order to understand front-running attacks, it is necessary to understand the pricing mechanism of DEX. DEXs work differently than traditional centralized exchanges, and most DEXs determine prices through an automated market maker (AMM) model [4] instead of a traditional order book. Unlike centralized exchanges, DEXs operate by relying on liquidity provided by users rather than the exchange itself. In the AMM model, the transaction price is determined according to the percentage of tokens deposited in the liquidity pool. A representative price determination algorithm is Constant Product Market Makers(CPMM) used in uniswap, which uses (1), a constant multiplication formula, where x and y represent the amount of two tokens in the pool, and k is a constant. At this time, the price is calculated by ensuring that the product of the quantities of the two tokens in the pool maintains a constant value. At this time, the larger the value of k (more liquidity providers), the smaller the price fluctuation that occurs when executing a transaction, providing good trading conditions with low slippage. However, these systems are exposed to risks such as price fluctuations and LP sandwich attacks.

$$x * y = k \tag{1}$$

LP Sandwich attack is the most common front-running-based attack in the DEX ecosystem. Fig. 2 shows the LP Sandwich attack sequence. The attack process is described in detail as follows.

i. Detection: The attacker monitors the blockchain mempool and finds a large pending transaction that is expected to significantly change the price of the liquidity pool.

ii. Attacker's front-running transaction: Before the victim's transaction is processed, the attacker places a purchase order for the same token, and the token price is artificially increased by the AMM.

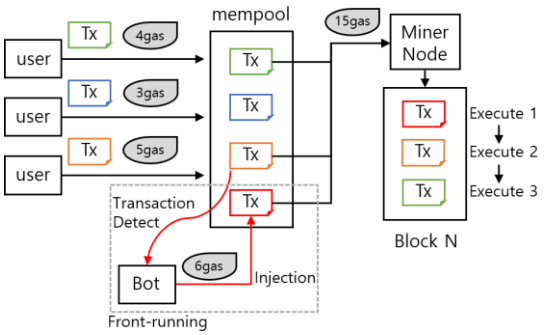

Fig. 1. Block generation process and front-running attacks.

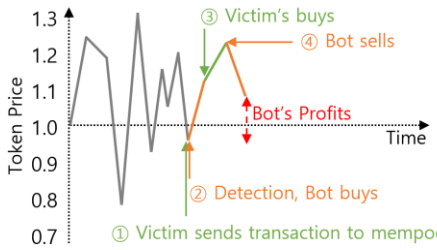

Fig. 2. Token price change due to MEV-Bot's LP Sandwich attack.

iii. Victim's Transaction: As the victim's transaction is processed, the token price is further increased based on the size of the transaction.

iv. Attacker's profit realization: The attacker sells the tokens at the increased price immediately after the victim's transaction and realizes a profit.

Due to this attack, users purchase tokens at a price higher than the price predicted when creating the transaction, causing damage. The liquidity provider sells tokens to the attacker at a low price and then purchases tokens at a high price, incurring a loss equal to the difference. This is an act that destroys the reliability of the platform by abusing the transparency that supports the reliability of the DEX and creating unfair transactions. MEV-Bot is carrying out these attacks on all transactions above a certain size that occur on DEX, and can be seen as forcing users to suffer losses [5]. In order to revitalize and expand decentralized exchanges, we regard the reckless profit-making behavior of these bots as an attack, and pursue defense against such strategic behavior.

C. *Related Work*

The traditional commit-reveal scheme is one of the methods that can be used in DEX to prevent frontrunning attacks. This method processes transaction information in two stages: commit (sending the hash value of the actual transaction) and reveal (revealing the actual transaction information). In the commit stage, the hash of the transaction is first transmitted to the network, and after a certain period of time, the actual transaction information is revealed in the reveal stage, preventing other users from abusing the transaction before it is disclosed. This increases the security and reliability of transactions, but includes problems such as delay in transaction processing and increased complexity of user experience. Research is being attempted to solve these problems [6][7], but it shows limitations in applying to the current blockchain DEX by modifying the entire blockchain protocol. In addition, it does not address general problems such as increased complexity of user experience [8][9]. Research such as submarine send, a modification of the traditional commit-reveal scheme, has also been proposed [10] [11]. Submarine send uses a method of randomly generating a completely new address on the network, sending an asset to that address, and then releasing and transmitting the asset during the revitalization process through the CREATE2opcode. This is difficult to track asset flows, but there are implementation difficulties, and there are difficulties in supporting ERC-20 tokens.

In order to solve the problem of the fee burden for double transmission of users' transactions in this commit-reveal scheme, research on meta-transactions that allow gasless transactions through a relay server was proposed [12]. In addition, an Ethereum solution was proposed to prevent disclosure of transaction details before transaction execution through a private mempool and to generate blocks through fee

bidding by affiliated validators [13][14]. However, these methods undermine the decentralization of the DEX as a server-based service is responsible for transmitting transactions in the middle, and are vulnerable to hacking and corruption of the central server. Other approaches include proposals to encrypt transactions sent to the mempool or to solve them through a layer 2 solution. However, not only are there limitations in application at the current stage, but inconveniences arise such as changes to the structure of the existing DEX and the need for a user-level node program.

III. Decentralized Commit-Reveal Scheme

We propose a new commit-reveal scheme to enhance user experience and decentralization, which were problems in previous research. The proposed scheme in this paper is based on Ethereum Uniswap V3. In the proposed scheme configuration, UserA is the transaction party, and STD (Swap Transaction Data) refers to the actual transaction data structure (data for swap transactions) for UserA to perform transactions with the DEX. The detailed information of STD is shown in Table 1. STD includes *{amountIn, amountOut, path[tokenX/tokenY], to, Dexpooladdr}* for transaction execution. The STD structure refers to the actual data used in *swapExactTokensForTokens()*, a smart contract function for transactions in the Uniswap protocol engine.

TABLE I. The Swap Transaction Data(STD) structure

Name	*Type*	*Description*
amountIn	uint	The amount of input tokens to send.
amountOut	uint	Minimum amount of output tokens that must be received to not revert a transaction.
Path	address[] calldata	An array of the token address. A pool must exist for each consecutive address pair and liquidity must exist.
To	address	Recipient of the output tokens.
Dexpooladdr	address	The address of the DEX token pair pool that will perform the actual transaction.

$$Proof := eth\text{-}sha3(STD \,||\, Nonce) \quad (2)$$

Proof refers to hashing data for the security of transaction information commit. Proof is structured as in (2), and each data is concated and hashed through the keccak-256 hash function, a sha3 function used in Ethereum. Through this, the transaction destination DEX and transaction data are not disclosed at the commit stage, which is the transaction proposal stage. It also eliminates the possibility of an attacker's brute force attack through random nonce.

The relay router is a smart contract for the operation of the actual commit-reveal scheme in the proposed scheme, and the overall structure of the scheme is shown in Fig. 3. This contract imports the ERC-20 token interface, which allows the router to be used as a Blockchain wallet. Additionally, the user grants the router permission to use the token through the approve function. Through this, the router can process swap actions according to the commit-reveal scheme through the *transferFrom()*. Users who wish to submit transactions with the DEX through Commit must provide the router with a minimum amountInMax allowance for input tokens before the reveal step, and the asset is locked until the deputy reveal step. The proposed commit-reveal scheme consists of Proof commit, STD reveal, and deputy reveal, and the operation of the scheme is shown in Fig. 4. Fig. 4 shows that UserA's actual

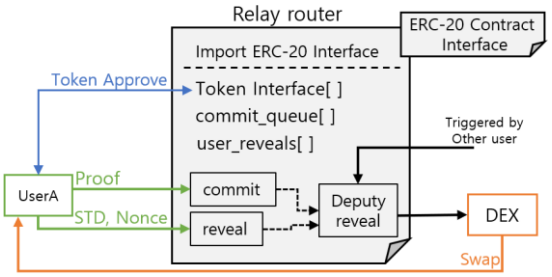

Fig. 3. Overall architecture diagram of the proposed scheme.

transaction is delivered to the DEX through deputy reveal, which is performed through UserA's commit-reveal participation and UserB's triggering.

Deputy reveal was proposed to prevent MEV-bot's front-running attacks that can be attempted by detecting STDs that users reveal at the STD reveal stage. When a user submits a commit for a transaction proposal, a Proof is created through (2), which prevents Proof-based MEV-bot attacks. Additionally, it prevents metadata-based inference by making it impossible to confirm the target DEX and swap transaction volume. The router contract sequentially stores proof commits sent from users through the commit queue, verifies actual transaction data at the STD reveal stage, and performs actual transactions through deputy reveal through a third party. After performing Proof commit, the user transmits STD reveal at an interval of t_1 time. The user's STD reveal operates as an actual transaction on the DEX through the deputy reveal, which is triggered along with the third party's STD reveal.

Proofs recorded through UserA's proof commit at N-t_1 of the scheme's operation timeline are recorded in order in the commit queue. In N, verification is performed on the STD and Nonce, which are the original data of the proof transmitted through UserA's STD reveal. Verification is performed by checking whether UserA's Proof in N-t_1 and the hash value according to (2) of the STD and Nonce disclosed by UserA in N are the same. Proof commit and STD reveal limit immediate reveal by an attacker through the minimum waiting time of t_1. The router blocks transmission transactions exceeding the minimum *amountIn* amount of the token associated with the user account for the verified STD reveal. Commit and reveal include the user account signature, and are verified and executed in the contract through *msg.sender*.

Deputy reveal is triggered for every STD reveal of a random user, and performs the actual transaction of the past k^{th} Proof in the commit queue. For example, in N+t_2, deputy reveal, which is triggered with the STD reveal of UserB, a third-party user of the scheme, checks UserA's Proof and STD pending in the commit queue and operates the actual

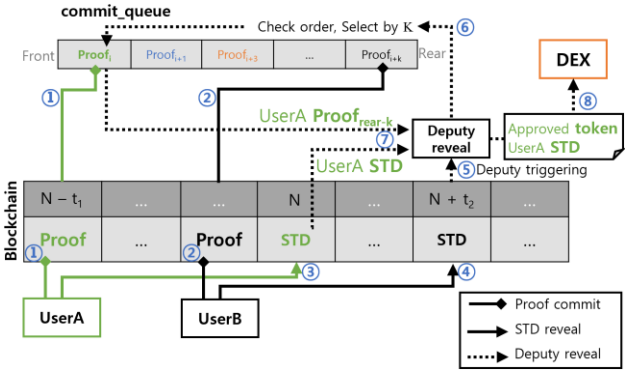

Fig. 4. Process of deputy reveal for decentralized commit-reveal scheme.

transaction. The user's transaction whose STD reveal has been confirmed waits for the addition of k other Proof data in the commit queue. UserA's Proof when k is met is performed in the order of actual transaction requests without frontrunning through the commit queue. A waiting time is given to UserA's Proof through k to prevent deputy reveal from being performed before the STD is submitted. At this time, k is variably adjusted depending on the network situation and the transaction volume of the DEX. Transactions proposed to DEX as deputy reveal along with UserA's approved token are performed according to the transaction request specification of the STD structure.

Router architecture and deputy reveal complement the security of the user's commit. In addition, transactions through DEX do not significantly change the structure of the existing decentralized financial platform, and enable sequential transactions without changing the blockchain protocol, preventing unfair transactions through front running. In addition, the possibility of an attack is reduced by imposing a risk on asset liquidity for MEV-bot attacks through asset locking after the STD reveal but before the deputy reveal stage. The proposed decentralized commit-reveal scheme can interact regardless of the structure of the DEX by processing commit and reveal through a smart contract. Additionally, it blocks malicious actions by middle managers by not using off-chain scripts or centralized services for the actual transactions to be processed.

IV. Security Analysis

We perform scenario-based verification to analyze the security of the proposed decentralized commit-reveal scheme. Through scenario-based verification, we confirm that the commit-reveal scheme based on relay router and deputy reveal can perform safe transactions and is resistant to MEV-bot for LP Sandwich attack through front running. The attack scenario is structured as shown in Fig. 5, and each scenario consists of an attack method at each point in time according to the user's scheme flow. The scenario configuration environment is as follows. The scenario environment is as follows: Users using the proposed commit-reveal scheme submit a Proof commit purchase order to purchase tokens to the blockchain through a relay router. Afterwards, users perform swap transactions on DEX according to the scheme flow through STD reveal and deputy reveal. The scenario assumes an Ethereum network using a common mining protocol, where all block validators prioritize transactions offering the highest gas. Additionally, it is assumed that all DEX users participate in swap transactions through the relay router.

Attack Scenario 1: LP Sandwich MEV-bot through traditional front-running confirms token transactions through mempool observation at the proof commit stage and attempts an attack by submitting a preceding transaction for a large transaction that can realize profits at a high gas price.

Defense: In the proposed scheme, the proof submitted at the Proof commit stage is hashed through (2) to eliminate the risk due to observation. Additionally, the address and transaction volume of the target DEX are not disclosed, preventing metadata-based inference. An attacker's replay attack to check Proof information is prevented through the transaction Nonce of the STD hashing process.

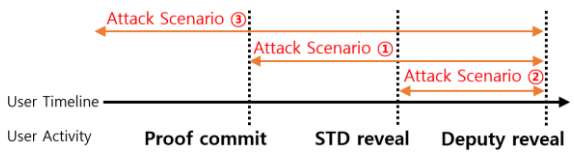

Fig. 5. Attack scenarios according to scheme flow.

Attack Scenario 2: MEV-bot fails to obtain information at the proof commit stage, and then verifies the actual data at the STD reveal stage and transmits a preceding transaction for a transaction that is certain to be profitable.

Defense: In order to conduct an actual transaction in the proposed scheme, the transaction request must be hashed at the commit stage and provided to the router. In Fig. 4, even if the attacker first checks the STD reveal at time N and participates in commit-reveal, the commit submitted at time N must submit the STD reveal at $N+t_1$ after the minimum waiting time of time t_1. Even if an attacker submits data before the user's STD or submits an attack-commit for an attack after randomly generating k commits, it is executed sequentially according to the commit queue at the deputy reveal stage where the actual transaction is performed.

Attack Scenario 3: Through the previous scenario, MEV-bot's front-running-based attack failed. Accordingly, the bot predicts transactions that will occur in the future, submits a random commit in advance, and waits for the attack target to occur. The attacker submits multiple commits to predicted points where large transactions are expected to occur.

Defense: If an attacker attempts to perform multiple commits and reveals, the attacker's assets are frozen during the period from STD reveal to deputy reveal, making the attack volume extremely small. Assume that there is a $Proof_{atk}$ submitted by an attacker predicting the STD_{user} before the user's $Proof_{user}$ is created at time N-1, and that the user's $Proof_{user}$ is submitted at time N. In this case, if the victim's STD_{user} is submitted first to $N+t_1$, the attacker can observe the transaction and a front-running attack may occur. However, to do this, the attacker must perform many proof commits, and the gas burden for executing the attack rises rapidly. Additionally, malicious commits can be limited by limiting the maximum number of commits. In the case of the attacker's LP Sandwich process, purchase and sale are carried out immediately without an intermediate buyer, and if a large amount occurs before the attacker's transaction. There is a significantly high possibility of loss of attack funds due to gas consumption, DEX fees, and forced transaction execution through deputy reveal.

V. Conclusion

This paper presents a new approach to solve the problem of front-running attacks occurring in decentralized exchanges. The proposed decentralized commit-reveal scheme overcomes the limitations of existing methods and effectively blocks attacks while maintaining the decentralized spirit and user experience of DEX. This paper provides basic knowledge about blockchain and front-runing, analyzes existing defense mechanisms, and explains in detail the structure and operating principles of the new scheme. In addition, we verified the stability of the proposed scheme through various scenarios and proved that transaction security in the DEX environment can be strengthened. This study is expected to contribute to providing a fair and transparent trading environment for all DEX users and reducing abnormal fee surges in blockchain networks. In addition, it improves platform reliability by blocking non-ideal transaction variables in user transactions.

REFERENCES

[1] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, "High-Frequency Trading on Decentralized On-Chain Exchanges," in *2021 IEEE Symposium on Security and Privacy (SP)*, May 2021, pp. 428–445.

[2] Z. Stucke, T. Constantinides, and J. Cartlidge, "Simulation of Front-Running Attacks and Privacy Mitigations in Ethereum Blockchain," in *34th European Modeling and Simulation Symposium, EMSS 2022*, Caltek, 2022, p. 041.

[3] "Analyst: Suspicious Bitcoin Mempool Activity, Transaction Fees Spike to $16," Cointelegraph. Accessed: Dec. 01, 2023. [Online]. Available: https://cointelegraph.com/news/analyst-suspicious-bitcoin-mempool-activity-transaction-fees-spike-to-16

[4] "Introducing Uniswap v3," Uniswap Protocol. Accessed: Dec. 01, 2023. [Online]. Available: https://blog.uniswap.org/uniswap-v3

[5] J. Fábregas, "Tracking Ethereum blockchain crypto attackers: Measuring sandwich attacks," Tarlogic Security. Accessed: Dec. 01, 2023. [Online]. Available: https://www.tarlogic.com/blog/ethereum-blockchain-sandwich-attacks/

[6] H. Zhang, L.-H. Merino, Z. Qu, M. Bastankhah, V. Estrada-Galinanes, and B. Ford, "F3B: A Low-Overhead Blockchain Architecture with Per-Transaction Front-Running Protection." arXiv, Sep. 05, 2023.

[7] M. A. Alturki and G. Roşu, "Statistical model checking of RANDAO's resilience to pre-computed reveal strategies," presented at the Formal Methods. FM 2019 International Workshops: Porto, Portugal, October 7–11, 2019, Revised Selected Papers, Part I 3, Springer, 2020, pp. 337–349.

[8] M. Arulprakash and R. Jebakumar, "Commit-reveal strategy to increase the transaction confidentiality in order to counter the issue of front running in blockchain," *AIP Conference Proceedings*, vol. 2460, no. 1, p. 020016, Aug. 2022.

[9] A. Canidio and V. Danos, "Commit-Reveal Schemes Against Front-Running Attacks (Extended Abstract)," in *DROPS-IDN/v2/document/10.4230/OASIcs.Tokenomics.2022.7*, Schloss-Dagstuhl - Leibniz Zentrum für Informatik, 2023.

[10] L. Breidenbach, P. Daian, F. Tramèr, and A. Juels, "Enter the Hydra: Towards Principled Bug Bounties and Exploit-Resistant Smart Contracts." 2017. Accessed: Dec. 01, 2023. [Online]. Available: https://eprint.iacr.org/2017/1090

[11] "To Sink Frontrunners, Send in the Submarines," Hacking Distributed. Accessed: Dec. 01, 2023. [Online]. Available: https://hackingdistributed.com/2017/08/28/submarine-sends/

[12] F. Darmawan and B. Bakhtiar, "PEMBANGUNAN GAME LOST CHAIN MENGGUNAKAN BLOCKCHAIN DAN GASLESS TRANSACTION," *INFORMATION SYSTEM FOR EDUCATORS AND PROFESSIONALS : Journal of Information System*, vol. 8, no. 1, pp. 95–106, Jul. 2023,

[13] "Overview | Flashbots Docs." Accessed: Dec. 01, 2023. [Online]. Available: https://docs.flashbots.net/flashbots-auction/overview

[14] X. Lyu, M. Zhang, X. Zhang, J. Niu, Y. Zhang, and Z. Lin, "An Empirical Study on Ethereum Private Transactions and the Security Implications." arXiv, Aug. 04, 2022.